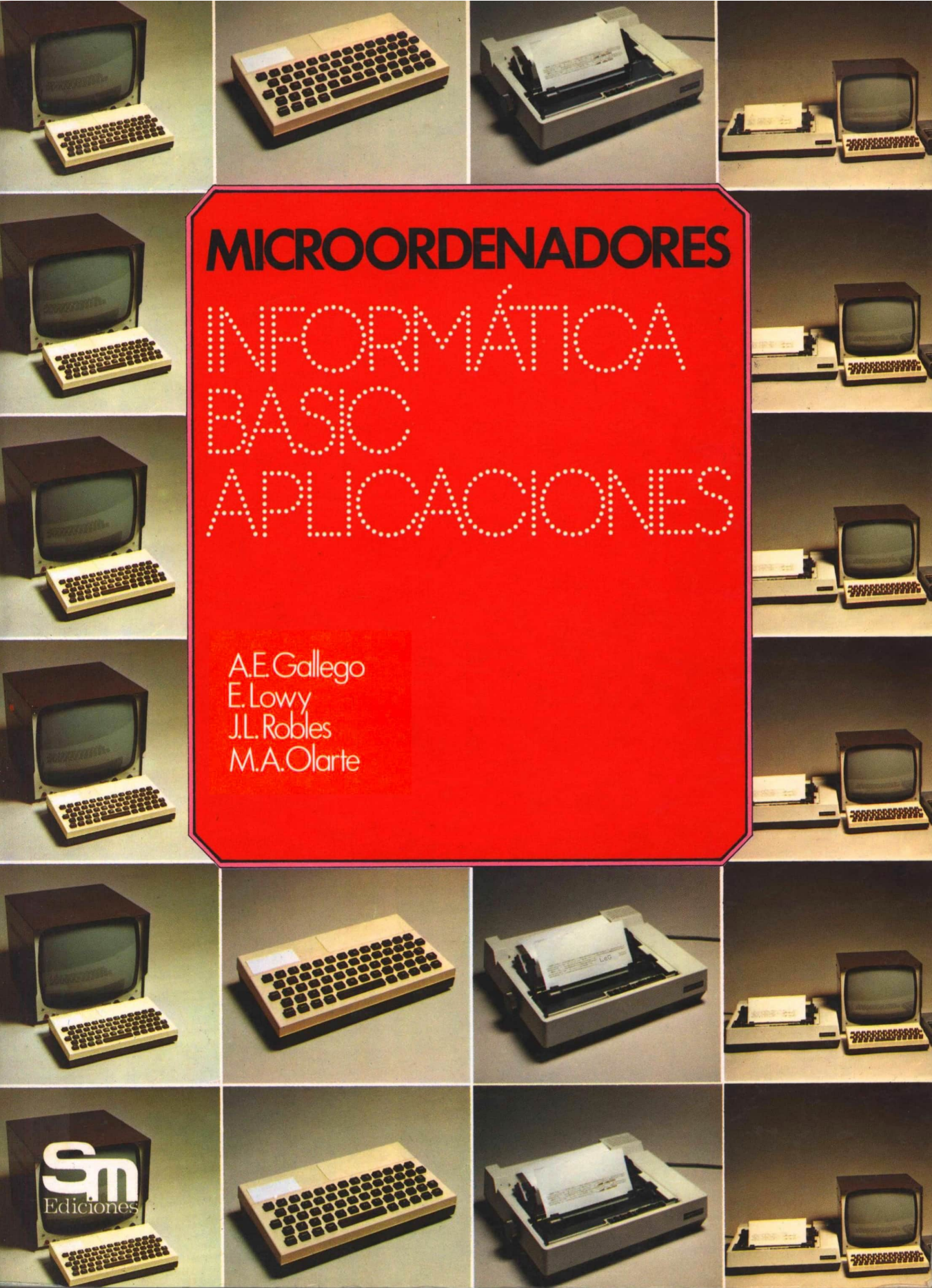


MICROORDENADORES

INFORMÁTICA
BASIC
APLICACIONES

A.E. Gallego
E. Lowy
J.L. Robles
M.A. Olarte

SM Ediciones

Coordinación: Serafín Mansilla Romo.

Autores: Antonio Enrique Gallego Palomero.
Agregado de Matemáticas de I. B.
Ernesto Lowy Frutos.
Agregado de Física y Química de I. B.
José Luis Robles Cid.
Licenciado en Ciencias Físicas.
María A. Olarte Madero.
Catedrática de Física y Química de I. B.

Maqueta: José Ugarte.

Dibujos: Modesto Arregui.

Fotografía: José Manuel Navia.
Archivo S.M.

Portada: Alfonso Ruano.

MICROORDENADORES

INFORMÁTICA BASIC APLICACIONES

A.E. Gallego
E. Lowy
J.L. Robles
M.A. Olarte

SM Ediciones General Tabanera, 39 - MADRID-25

© EDICIONES S. M. - Madrid / I.S.B.N.: 84-348-1254-1 / Depósito legal: M. 34.513-1983
Fotocomposición: SECOMP - Madrid / Imprime: Gráficas VELASCO, S. A. - Antonio de Cabezón, 13 - Madrid-34
Impreso en España - Printed in Spain

INDICE

Pág.

PROLOGO 7

1. MICROORDENADORES 9

Una máquina para manejar información • Ordenadores y microordenadores • Mecanismos en el tratamiento de la información • La memoria • Unidades aritmética y de control • Entrada y salida de información • Todo lo que un microordenador puede hacer • El lenguaje BASIC.

2. EMPEZANDO A PROGRAMAR EN BASIC 17

Un programa en BASIC • Cómo procesar un programa • Cómo cambiar una instrucción en un programa • Errores de sintaxis • Instrucción REM • Instrucción PRINT • El «punto y coma» y la «coma» en BASIC • Uso del PRINT para imprimir cálculos directamente • Escritura de los números en lenguaje BASIC • Ejercicios de recapitulación.

3. SALTOS INCONDICIONALES Y CONDICIONALES 32

Programa para escribir infinitas veces una frase • Programa: obtener los múltiplos de 3 hasta un cierto número • Programa: recuento de alumnos que miden entre 1,60 y 1,70 • Relaciones • Programa: calcular valores medios • Para parar la ejecución de un programa: la instrucción STOP • Ejercicios de recapitulación.

4. DATOS DE UN PROGRAMA 42

Instrucciones READ y DATA • Instrucción RESTORE • Ejercicios de recapitulación.

5. BUCLES 47

Bucles en un programa • Bucles con incremento distinto a 1 • Bucles en el interior de otros bucles (bucles anidados) • Ejercicios de recapitulación.

6. CADENAS 56

Cadenas de caracteres • Suma o concatenación de cadenas • Longitud de una cadena • Extracción de subcadenas • Comparación de cadenas • Programa: cambio de moneda • Ejercicios de recapitulación.

7. LISTAS Y TABLAS 65

Listas (vectores) • Ordenando números • Verificación del programa • Tabla de dos dimensiones (matrices) • Programa: suma de las filas y columnas de una tabla • Tablas de caracteres alfanuméricos • Ordenación de cadenas • Ejercicios de recapitulación.

Pág.

8. FUNCIONES Y SUBRUTINAS 79

Algunas funciones del lenguaje BASIC • Aplicación: cambio de unidades • Funciones trigonométricas, exponenciales y logarítmicas • La función azar • Aplicación: simulación de experimentos aleatorios • Definición de funciones • Subrutinas • Programa: comprobación del resultado de la suma y del producto de dos números • Ejercicios de recapitulación.

9. GRAFICOS 93

Instrucciones PRINT TAB • Instrucción PRINT AT (o PLOT) • Gráficos en alta resolución • Ejercicios de recapitulación.

10. INTRODUCCION A LAS APLICACIONES 101

Algunas pautas para resolver problemas con microordenadores • Objetivo de las aplicaciones.

11. MATEMATICAS 107

Resolución de la ecuación de segundo grado • Aprendiendo a multiplicar • Anualidades de amortización • Suma de matrices • Producto de matrices • Resolución de sistemas de ecuaciones lineales • Operaciones con vectores • Módulo y cosenos directores de dos vectores.

12. REPRESENTACION GRAFICA DE FUNCIONES 125

Programa para dibujar gráficas de funciones • Gráficos en alta resolución • Figuras de Lissajous.

13. CIENCIAS 132

Análisis del concepto de velocidad instantánea (derivada de una función en un punto) • Cálculo de la aceleración de un sistema de masas • Análisis del movimiento de un cohete de masa variable • Cálculo del período de revolución de un planeta • Análisis de las ondas de una cuerda • Estudio de las interferencias • Estudio de las oscilaciones de un circuito eléctrico • Simulación de la desintegración radiactiva • Utilización del sistema periódico de los elementos • Igualación de ecuaciones químicas • Valoración ácido fuerte-base fuerte • Cálculo del número de células nacidas en un determinado tiempo • Un problema de ecología • Estudio de leyes genéticas.

14. GEOGRAFIA E HISTORIA 161

Averiguación de capitales de países • Pirámides de población • Ordenar cronológicamente.

15. LENGUA E IDIOMAS 169

Ejercicios de acentuación • Conjugación de verbos • Identificación de preposiciones inglesas • Repasando la morfología y sintaxis latinas.

16. OTRAS APLICACIONES 178

Utilización de una pequeña biblioteca • Control de la situación académica de alumnos.

APENDICE 187

BIBLIOGRAFIA 189

Prólogo

Los ordenadores, tal como hoy día los concebimos, han aparecido hace menos de cincuenta años, pero su difusión en tan corto intervalo de tiempo ha sido espectacular. En el momento presente, y sin temor a equivocarnos, podemos decir que vivimos rodeados de ordenadores, ya que, en mayor o menor grado, detrás de cualquier actividad de nuestro entorno se encuentra una de estas máquinas.

Consecuentemente con esta realidad, que algunos han dado en llamar la «revolución informática», los ordenadores también se han introducido en el campo de la enseñanza. Ya son muchos los centros, en nuestro país, que imparten enseñanzas de programación con microordenadores personales, al mismo tiempo que los utilizan como ayuda didáctica en otras disciplinas.

El libro que presentamos recoge nuestra experiencia como docentes sobre todo en el área de ciencias, aunque gracias a la ayuda prestada por compañeros de otras áreas hemos podido extender la aplicación de las técnicas informáticas a otras disciplinas.

El lenguaje de programación adoptado es el BASIC, por ser el que utilizan la mayoría de los microordenadores actualmente presentes en el mercado y por ser suficientemente sencillo para poder ser utilizado a nivel de BUP, COU y FP.

La primera parte del libro está dedicada a introducir en forma gradual las instrucciones BASIC, siempre a través de programas de progresiva complejidad, los cuales se pueden ir probando en cualquier microordenador.

En una segunda parte se dan programas de aplicación múltiple que, en nuestra opinión, pueden ser una buena herramienta para el quehacer docente, pues al mismo tiempo que ayuda al alumno a aprender el lenguaje de programación, le motiva en la asignatura en la que se hace la aplicación correspondiente.

Esperamos que el libro sea de utilidad para los que se inician en el campo de la programación con microordenadores que utilizan lenguaje BASIC, y también sirva de ayuda para aquellos profesores que deseen aplicar los medios informáticos al ámbito de la enseñanza.

LOS AUTORES

Todos los programas de este libro se han ejecutado en diversos microordenadores. Pero dado que actualmente existen tantos en el mercado, es posible que algunos programas necesiten un pequeño ajuste para ejecutarlos en algún «micro» determinado.

1 Microordenadores

Una máquina para manejar la información

El concepto tradicional de máquina, polea, torno, etc., ha experimentado una evolución. En la actualidad, llamamos máquina a todo dispositivo que ayuda a satisfacer nuestras necesidades: la de viajar más rápido (automóviles, aviones, etc.), la de comunicarnos mejor (radio, televisión, teléfono, etc.) y otras muchas más.

Una de estas necesidades, que ha surgido como consecuencia de la gran complejidad alcanzada por la ciencia, por la economía e incluso por nuestra vida ordinaria, es la de *mecanizar* una parte del trabajo intelectual. Esta mecanización supone *organizar, almacenar* y *procesar* (*) los datos que llegan desde cualquier parte y siempre en grandes cantidades.

Las máquinas que ayudan a realizar esta función son los *ordenadores*. Han aparecido paralelamente al desarrollo de la electrónica y con ellos se consigue almacenar, reproducir y tratar la información en condiciones que superan la propia capacidad humana.

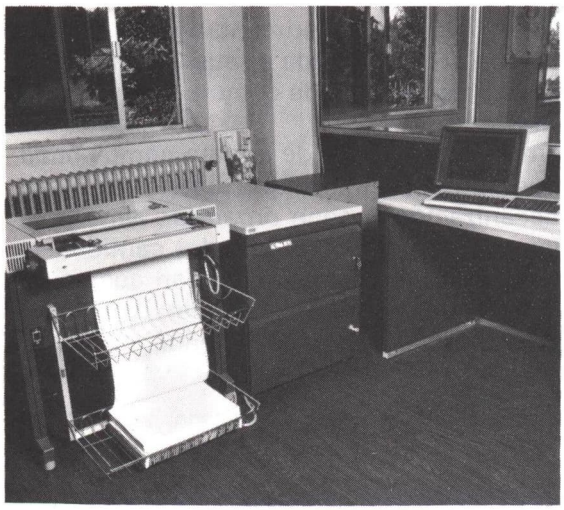
(*) *Procesar:* Acción que realiza un microordenador para interpretar un programa y ejecutarlo.

Un ordenador puede realizar muchos cálculos por segundo, puede almacenar gran cantidad de información y reproducirla en muy poco tiempo. Además, todo ello, sin cansarse ni distraerse, aunque lo que tenga que realizar se repita muchas veces.

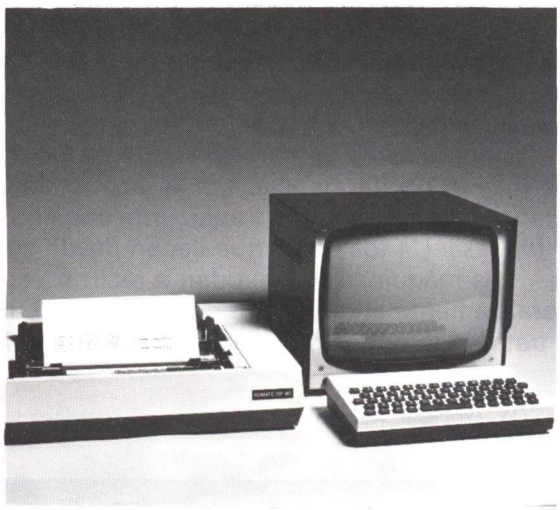
Sin embargo, un ordenador no sirve para nada si no se le *programa correctamente,* es decir, si no se le dice lo que tiene que hacer. El ordenador, como toda máquina, no piensa; hay que pensar por él.

Ordenadores y microordenadores

¿Qué diferencia hay entre un ordenador y un microordenador? En la actualidad, no existe una diferencia clara entre ambos tipos de máquinas, pues las funciones que realizan las dos son esencialmente las mismas.



Miniordenador



Microordenador

El tamaño puede ser un elemento diferenciador, aunque tampoco definitivo, pues dentro de los microordenadores pueden haber de tamaño tan pequeño como para que podamos cogerlos con una mano o bien de tamaño tal que necesiten ser colocados encima de una mesa de despacho.

En general, la característica que más diferencia al ordenador del microordenador está en que el primero tiene una mayor capacidad para almacenar y procesar información y, además, realizar el trabajo en menos tiempo.

Por esto, los ordenadores se utilizan cuando la información es muy extensa y compleja; por ejemplo, para controlar la reserva de billetes de ferrocarril de todo un país. Los microordenadores, por el contrario, se emplean para hacer tareas más sencillas y próximas a nuestra vida ordinaria; por ejemplo, para llevar la contabilidad de una pequeña empresa, para controlar la actividad de un trabajo profesional (médicos, ingenieros, asesorías, etc.). En lo sucesivo sólo nos ocuparemos de estos últimos.

Mecanismos en el tratamiento de la información

Para ver cómo trabaja un microordenador, veamos, con un ejemplo, cómo pueden mecanizarse ciertos procesos intelectuales.

Sea la expresión:

$$Y=0.3 * X+5 \qquad \text{para } X=1 \qquad (*)$$

— En la parte derecha del signo «igual» se describe el *tratamiento* que hay que hacer: un número (0.3) se multiplica por el valor que toma la variable X (X=1) y al resultado se le suma otro número (5).

— En la parte izquierda, en Y, se *almacena* el resultado (Y=5.3), el cual se suele escribir en el papel.

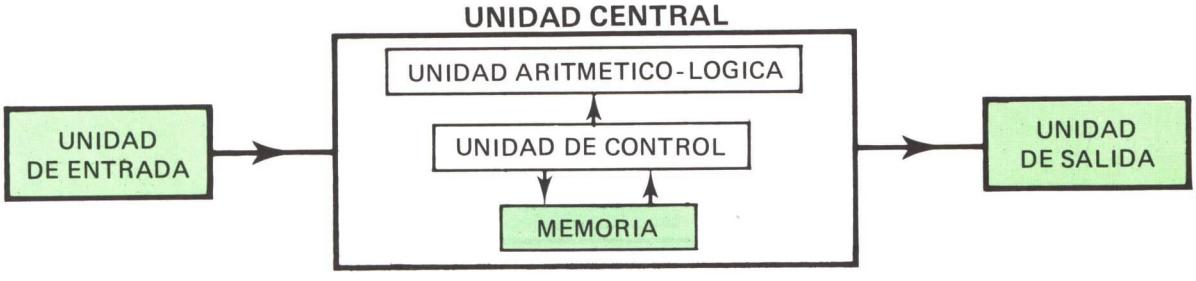
Todo este proceso intelectual el microordenador lo mecaniza en las siguientes etapas:

- Lectura del valor de la variable X.
- Lectura de la constante 0.3 y del signo operador «por». Interpretaciones de la expresión 0.3 * X para X=1 y cálculo.
- Lectura de la constante 5 y del signo (operador) +. Interpretación de 0.3 * X+5 para X=1 y cálculo.
- Escritura del resultado obtenido almacenado en Y.

En resumen, las tres funciones esenciales del proceso, son:

- **Lectura.**
- **Cálculo.**
- **Escritura.**
- Además de la función de **memorización** que tiene que hacer antes de la escritura, para retener la fórmula y los resultados intermedios y final.

Estas funciones las puede realizar un microordenador utilizando los dispositivos que se esquematizan en la figura.



— La lectura y escritura se realizan en los dispositivos de *entrada* y *salida*.
— La búsqueda de la fórmula, su interpretación y cálculo se realiza en la *unidad central* que incluye la *memoria* como una de sus tres partes.

(*) La notación 0.3 * X+5 es la que acepta el microordenador; equivale, en notación algebraica, a 0,3X+5. El signo de multiplicar es un asterisco. La letra O y el cero, 0, se encuentran en teclas distintas en el teclado del microordenador; el cero se distingue de la letra porque aparece «tachado» (Ø). En este libro no tacharemos el cero porque en tipos de imprenta ambos símbolos están perfectamente diferenciados (cero, 0; letra, O).

El microordenador lo único que hace es *seguir las instrucciones* que se le dan. El conjunto de instrucciones que necesita un microordenador para realizar una tarea determinada constituye un **programa.** Los programas deben ser específicos para cada tarea que se desee haga el microordenador y hechos de forma que los pueda entender y ejecutar. Así, para sumar dos números X e Y habría que escribir el programa formado por las instrucciones siguientes:

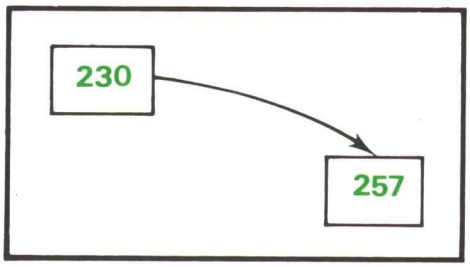
— Introduce el número X.
— Introduce el número Y.
— Suma los números X e Y y llama al resultado Z.
— Escribe el valor de Z.

La memoria

La memoria de un microordenador tiene como misión *almacenar* los datos para, posteriormente, ser utilizados. Cuando la memoria recibe información hace tres funciones: la registra, la conserva y la *devuelve* cuando se le pide.

1	2	3	4	5 **227**
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30

La memoria de un microordenador podría compararse con un panal formado por muchas celdillas llamadas *posiciones de memoria.* Al «memorizar» una información, cada posición de memoria puede contener una instrucción del programa, una parte de los datos o el resultado de algún cálculo parcial; por ejemplo, en la posición de memoria identificada con el número 5, está almacenado el dato 227.

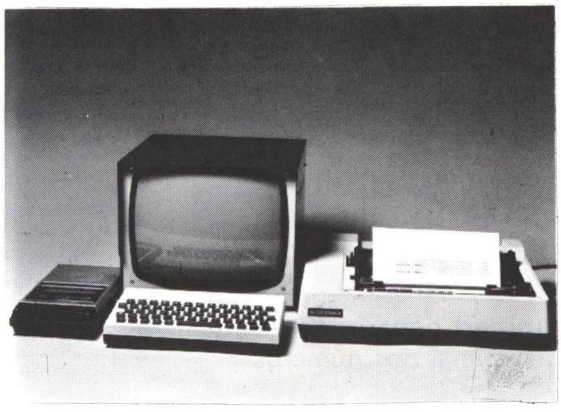


Muchas veces el microordenador necesita «llevar» el contenido de una posición de memoria a otra para poder ejecutar el programa.

En este caso es importante advertir que la posición de memoria de donde sale la información no pierde su contenido, pues todo ocurre como si lo que saliera de ella fuese una reproducción de lo que hay. Sin embargo, ocurre lo contrario con la posición de memoria a la que llega la información. Esta «olvida» lo que había antes del movimiento para «memorizar» lo que le acaba de llegar.

Así, el número 230 permanece, mientras que 257 desaparece, siendo sustituido por 230. Pasa algo parecido que con una cinta magnetofónica: cuando se reproduce el sonido no se borra, pero sí al grabarla.

Por último, como el tamaño de la memoria del microordenador es restringido, sólo puede almacenar el programa y los datos del programa que se está realizando en el momento.

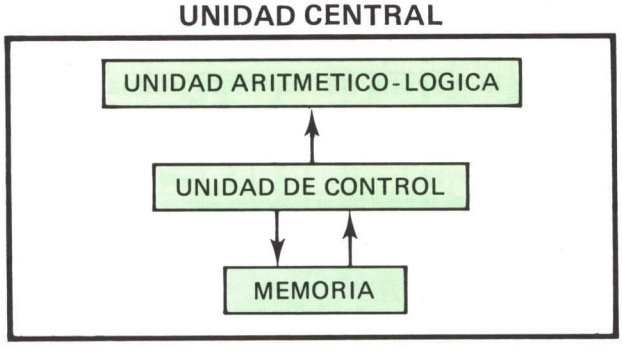


Memorias externas.

Pero, de la misma manera que una persona cuando tiene que marcar un teléfono lo busca en la agenda o en la guía telefónica, es decir, en una *memoria externa,* el que usa el microordenador puede almacenar información en dispositivos externos de memoria; por ejemplo, cintas magnéticas, disquetes, etc.

Unidades aritmética y de control

Además de la memoria, la **unidad central** de un microordenador tiene otras dos unidades: la **aritmético-lógica** y la de **control.** Estas unidades utilizan un conjunto de circuitos electrónicos especializados para cada función que tienen que realizar.



La unidad de control tiene como misión *identificar las instrucciones que se le dan a la máquina;* por ejemplo, para el programa anterior, éstas serían:

- Toma el valor de X.
- Multiplícalo por 0.3.
- Suma 5 al resultado obtenido.
- Coloca el resultado en Y.

La unidad aritmético-lógica *realiza las operaciones* que se indican en la instrucción. Esta unidad puede hacer *operaciones aritméticas* como:

— Sumar.
— Restar.
— Multiplicar.
— Dividir.

Y *comparaciones lógicas* como:

— 3 es menor que 4.
— A es anterior a B.
— Etc.

Para ver el funcionamiento conjunto de la **unidad central de proceso** estudiemos cómo el microordenador lleva a cabo la operación «sumar dos números».

Las etapas que sigue son:

1.º Almacena los dos datos en la memoria.
2.º La unidad de control saca los datos de la memoria y los coloca en la unidad aritmética.
3.º La unidad de control indica a la unidad aritmética que debe realizar la operación suma.
4.º La unidad de control indica a la aritmética que devuelva el resultado a una posición de memoria.

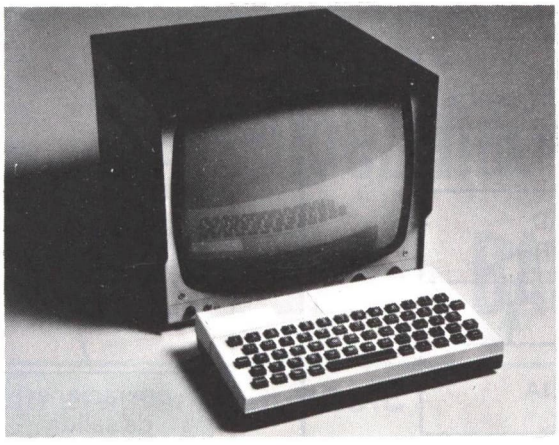
Es decir, la unidad de control es entonces como un «director» que controla el procesado de las instrucciones, encargándose también de desplazar los datos desde la memoria a la unidad aritmética, y luego desde ésta a la memoria.

Entrada y salida de información

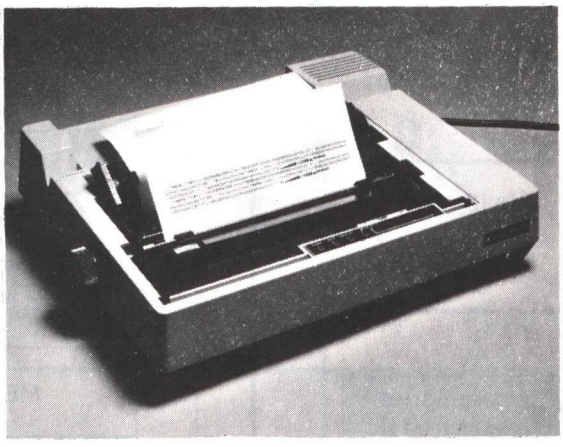
Un microordenador, además de la unidad central, tiene también dispositivos que le permiten comunicarse con el que lo va a usar. Para introducir las instrucciones y los datos en el interior de la máquina, la mayoría de los microordenadores disponen de un teclado como el de las máquinas de escribir, aunque en ocasiones la introducción de información se hace directamente desde dispositivos de memoria externos (disquetes, cintas).

Teclado de un microordenador.

Cuando se alcanza el resultado de una operación, éste se puede hacer aparecer en la pantalla (incluso muchas veces vale la de un televisor), imprimirlo sobre un papel o bien grabarlo, para utilizarlo en otro momento en un dispositivo de memoria externo.

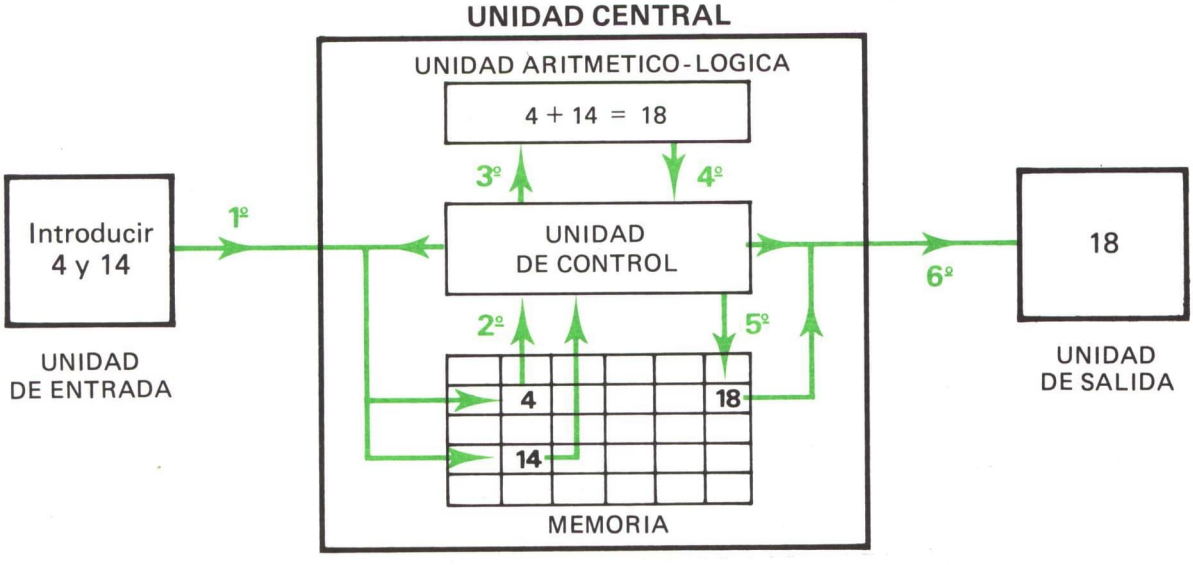


Pantalla de un microordenador.



Impresora de un microordenador.

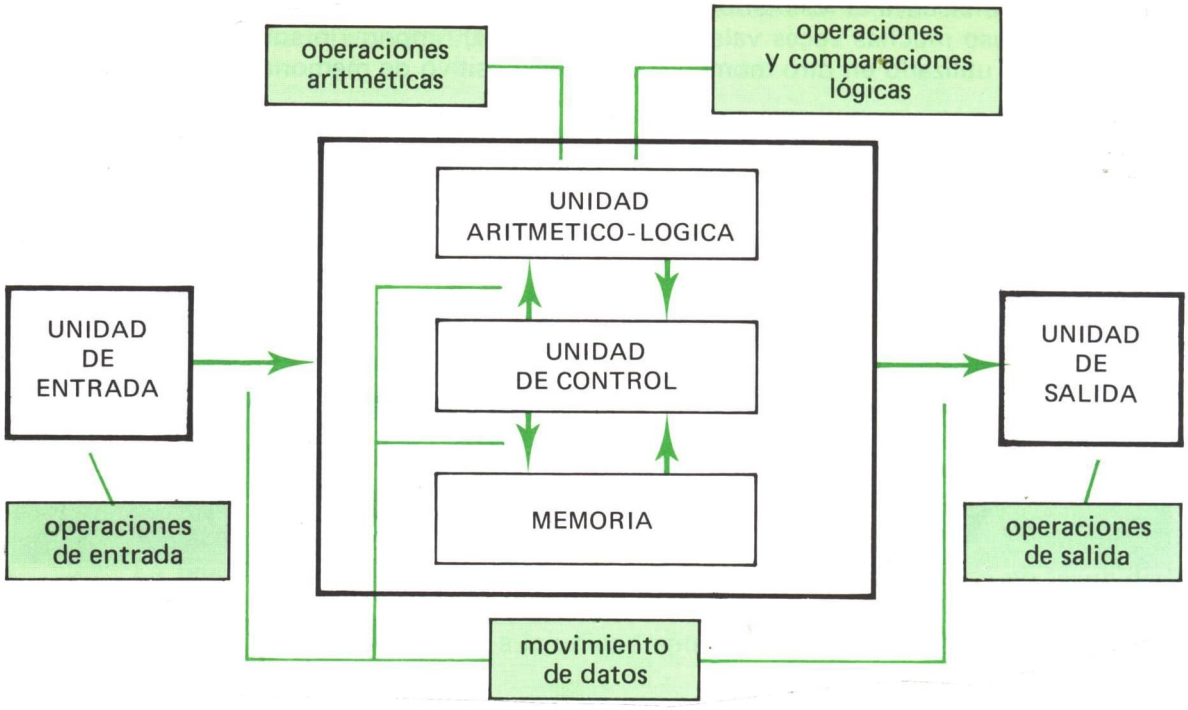
Analicemos los pasos anteriores en el caso concreto de la suma de los números 4 y 14.



Todo lo que un microordenador puede hacer

En resumen, lo que un microordenador puede hacer, se resume en lo siguiente:

- *operaciones de entrada y salida,*
- *operaciones aritméticas,*
- *comparaciones y operaciones lógicas,*
- *movimiento de datos.*



De la habilidad del usuario y de su ingenio depende de que estas máquinas ayuden a cosas como llevar hombres a la Luna, controlar el tráfico, diagnosticar enfermedades, etc., y todo utilizando un número tan restringido de posibilidades.

El lenguaje BASIC

Hemos señalado anteriormente que la información que se introduce en el microordenador debe dársele de tal forma que la máquina la «entienda». Si a una persona que sólo habla ruso se le comunica algo en inglés, la persona no entiende lo que se le está diciendo porque no se utiliza su propio lenguaje.

Lo mismo ocurre en los microordenadores. La información hay que dársela en un lenguaje adecuado. Un lenguaje que entienden la mayoría de los microordenadores es el lenguaje BASIC, palabra que deriva de la expresión inglesa «BEGINNER-s ALL-PURPOSE SYMBOLIC INSTRUCTION CODE». Este lenguaje utiliza algunas palabras en inglés, símbolos de puntuación habituales y funciones matemáticas corrientes.

En lenguaje BASIC, el programa para realizar la ecuación

$$Y=0.3 * X+5$$

se escribe:

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

Más adelante explicaremos detalladamente el significado de estos términos. Otros lenguajes utilizados también en ordenadores y microordenadores son: **Pascal, Cobol, Fortran, Forth, Logo,** etc.

2 Empezando a programar en BASIC

Un programa en BASIC

Analicemos con un poco detalle el programa en BASIC propuesto en el capítulo anterior, que consiste en:

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

(introducir por teclado el valor de X)
(efectuar estas operaciones y asignar el resultado a Y)
(escribir el valor de Y)
(fin)

Se observa:

1.º El programa tiene varias líneas numeradas de 10 en 10. Cada línea representa una instrucción y el número de línea sirve para precisar el orden en el que el ordenador debe ejecutar las instrucciones.

Se podrían haber numerado las líneas con los números 1, 2, 3 y 4, pero la numeración de 10 en 10 permite añadir en cualquier momento instrucciones olvidadas entre ellas.

2.º Cada instrucción comienza por una palabra o una expresión inglesa que indica el tipo de acción a ejecutar.

PRINT:	escribe, imprime
LET:	deja, haz
INPUT:	introduce
END:	fin

3.º Los caracteres que siguen a esta orden son símbolos que representan datos, variables u operadores.

Así, en la instrucción:

```
20  LET Y=0.3 * X+5
```

- 20 es el número de instrucción.
- LET es la instrucción en BASIC.
- Y y X son variables.
- 0.3 y 5 son números.
- =, + y * son operadores aritméticos.

Cómo procesar un programa

Para introducir el programa en el microordenador se escriben las instrucciones sobre el teclado *pulsando a continuación la tecla* [ENTER] (o [NEW-LINE] o [RETURN], etc., según el microordenador) *después de cada instrucción.* En este libro se utilizará [ENTER].

Sobre el teclado escribiremos:

```
10  INPUT X [ENTER]
20  LET Y=0.3 * X+5 [ENTER]
30  PRINT Y [ENTER]
40  END [ENTER]
```

Indicaciones para teclear:

— Al final de cada instrucción no olvidarse de accionar la tecla [ENTER] o su equivalente.

— La letra O y el número 0 se encuentran en teclas distintas.

Aunque en lo sucesivo no lo haremos, hemos incluido al final de cada instrucción el recuadro [ENTER] para insistir en que hay que pulsar dicha tecla después de cada instrucción. *Si esta tecla no se pulsa, el ordenador no se da por enterado de la instrucción escrita;* es decir, no pasa a la memoria.

Las instrucciones van apareciendo en la pantalla y al mismo tiempo almacenándose en la memoria.

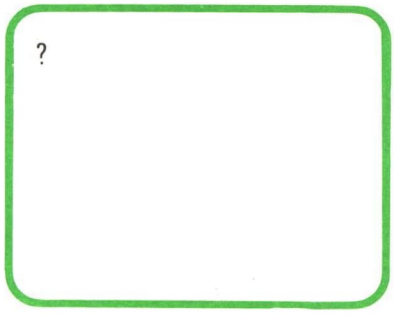
```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

Por ahora, el programa se encuentra almacenado en la memoria. ¿Qué tenemos que hacer para que lo ejecute, es decir, para que interprete las instrucciones y realice los cálculos?

Para que el microordenador *ejecute* el programa se escribe la palabra RUN pulsando a continuación la tecla [ENTER].

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

— Al ejecutar la primera instrucción aparece en la pantalla un signo de interrogación. (En algunas máquinas puede ser una L.)



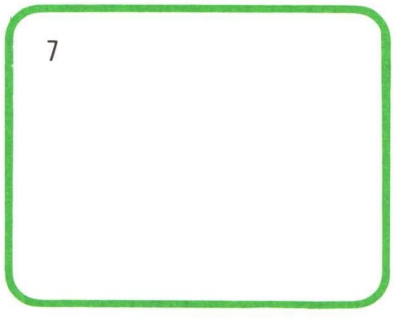
Con este interrogante (o con L) el microordenador pregunta qué valor deseamos asignar a X.

— La instrucción:

```
10  INPUT X
```

es una instrucción de *entrada de datos*. Cuando el microordenador encuentra esta orden interrumpe el programa para pedir al usuario la entrada de un valor de X.

Si, por ejemplo, damos el valor 7 y a continuación pulsamos la tecla ENTER, la ejecución del programa continúa y el número 7 se almacena en una posición de memoria representada por el símbolo X.



— La instrucción:

```
20  LET Y=0.3 * X+5
```

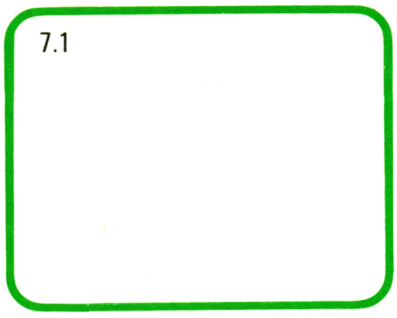
indica al microordenador que realice las operaciones aritméticas que se incluyen en la fórmula y que el resultado lo almacene en otra posición de memoria representada por el símbolo Y.

Si asignamos a X el valor 7, el microordenador almacena en Y el resultado del cálculo 0.3 * 7+5=2.1+5=7.1.

— La instrucción:

```
30  PRINT Y
```

imprime el valor que toma el símbolo Y, el cual aparece en la pantalla.



— La instrucción:

```
40  END
```

indica al microordenador que el programa se ha terminado.

En muchos microordenadores no hace falta colocar al final la instrucción END. En este programa han aparecido dos operadores, + y *, para sumar y multiplicar, respectivamente. Los operadores aritméticos que se suelen usar en lenguaje BASIC son los siguientes:

Paréntesis ( )
Elevar a una potencia: ↑ o * *
Cociente /
Producto *
Diferencia −
Suma +

Se han colocado de mayor a menor rango. Es decir, que en una expresión aritmética dada se realizan primero los paréntesis, a continuación las potencias, después los cocientes y productos y, por último, las diferencias y sumas.

Así, en resumen, para introducir un programa en el microordenador se teclean las instrucciones pulsando, después de escribir cada una, la tecla ENTER

```
10  INPUT X      ENTER
20  LET Y=0.3 * X+5      ENTER
30  PRINT Y      ENTER
40  END      ENTER
```

El programa aparecerá escrito en pantalla en la forma

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

Para ejecutarlo escribimos **RUN** pulsando a continuación la tecla [ENTER].

Aparece un interrogante en la pantalla y, después de escribir 7, pulsando a continuación la tecla [ENTER], aparecerá en pantalla 7.1.

Cómo cambiar una instrucción en un programa

Todos los microordenadores tienen una tecla para hacer correcciones cuando se cometen errores antes de introducir las instrucciones en la memoria (cuando no se ha pulsado todavía la tecla [ENTER]).

Sin embargo, si deseamos hacer algún cambio en el programa que está en la memoria del microordenador procederemos de la siguiente manera.

Primero debemos saber qué número corresponde a la instrucción o instrucciones que queremos modificar. Para ello debe aparecer en pantalla el *listado* del programa. Es decir, el programa tal como lo escribimos en el teclado.

Para obtener el listado basta con escribir en el teclado la instrucción LIST pulsando [ENTER] a continuación.

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PRINT Y
40  END
```

Si ahora queremos cambiar la ecuación que está en la instrucción 20 por otra, basta teclear la nueva instrucción, por ejemplo:

```
20  LET Y=21 * X+5    [ENTER]
```

Escribiendo LIST, y a continuación pulsando la tecla [ENTER], aparecerá en pantalla el programa modificado.

```
10  INPUT X
20  LET Y=21 * X+5
30  PRINT Y
40  END
```

También podemos borrar el contenido total de la memoria escribiendo NEW (nuevo programa), pulsando a continuación la tecla [ENTER]. También se conseguirá un borrado total desconectando y conectando de nuevo el microordenador, ya que los contenidos de la memoria se borran cuando no hay corriente eléctrica.

Errores de sintaxis

Cuando se teclea un programa pueden cometerse errores de escritura o de otros tipos, que iremos viendo más adelante.

Por ejemplo, el programa anterior podríamos haberlo escrito así:

```
10  INPUT X
20  LET Y=0.3 * X+5
30  PTINT Y
40  END
```

cometiendo un error en la instrucción 30.

En la pantalla aparecería (después de escribir RUN y a continuación [ENTER]).

```
? 30
SINTAX ERROR IN 30
```

El microordenador nos informa que hemos cometido un error, imprimiendo un mensaje que puede ser diferente en cada microordenador (puede ser incluso un número clave).

Para corregirlo deberemos modificar la instrucción en la que se ha detectado el error, usando el procedimiento visto en el apartado anterior.

LET (*)

Formato: LET [variable]=expresión.
Función: calcula el valor de la expresión y lo almacena en la variable.

(*) En la mayoría de los microordenadores no es necesario poner la palabra LET, basta escribir la variable; por ejemplo, la instrucción

```
20  LET Y=0.3 * X+5
```

la aceptan escrita así:

```
20  Y=0.3 * X+5
```

Ejercicios

2.1. Hacer un programa que calcule el producto de dos números e imprima el resultado.

2.2. Hacer un programa que calcule el cociente de dos números e imprima el resultado.

2.3. Hacer un programa que eleve al cuadrado un número e imprima el resultado.

2.4. Hacer un programa que calcule el cubo de un número e imprima el resultado

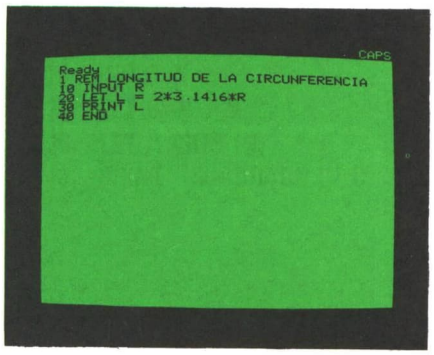
Instrucción REM

Con el siguiente programa el microordenador calculará la longitud de una circunferencia de radio R:

```
 1  REM LONGITUD DE LA CIRCUNFERENCIA
10  INPUT R
20  LET L=2 * 3.1416 * R
30  PRINT L
40  END
```

Para almacenarlo en memoria tecleamos cada línea seguida de [ENTER].
Para ejecutar el programa escribimos RUN y pulsamos [ENTER].
El microordenador pedirá un valor del radio (aparece en la pantalla un signo ?) y a continuación, una vez introducido éste, aparecerá el valor de la longitud de la circunferencia pedida.

Si a continuación listamos el programa (LIST-[ENTER])



vuelven a aparecer todas las instrucciones dadas. Sin embargo, la primera de ellas:

```
1  REM LONGITUD DE LA CIRCUNFERENCIA
```

no indica nada al microordenador; éste actúa como si no se le hubiese dado.

La instrucción **REM** *no supone acción alguna por parte del microordenador, pero sirve para poner título a un programa o a una parte de él.* Sin embargo, hay que tener en cuenta que esta instrucción sí ocupa posiciones de memoria.

Ejercicio

2.5. Escribir programas que calculen el área de un círculo, de un triángulo y de un cuadrado.

REM

Formato: REM lista de caracteres.
Función: añade aclaraciones a un programa sin afectar su ejecución.

Instrucción PRINT

El programa que calcula la longitud de una circunferencia puede mejorarse añadiendo las siguientes instrucciones:

```
 1  REM LONGITUD DE LA CIRCUNFERENCIA
 5  PRINT "INTRODUZCA EL RADIO"
10  INPUT R
15  PRINT R; "METROS"
20  LET L=2 * 3.1416 * R
30  PRINT "EL RESULTADO ES"; L; "METROS"
40  END
```

que hará aparecer en la pantalla

```
INTRODUZCA EL RADIO ←————————————— corresponde a la línea 5
? ←————————————————————————————— corresponde a la línea 10 (introducimos 5)
5 METROS ←————————————————————— corresponde a la línea 15
EL RESULTADO ES 31.416 METROS ←————— corresponde a la línea 30
```

En la línea 30 se ven algunos ejemplos de los elementos que pueden incluirse detrás de la palabra PRINT:

— Un texto entrecomillado, como "EL RESULTADO ES" y "METROS"; se imprime en la pantalla exactamente igual pero sin comillas.

— Una variable, como L; se imprime su valor (31.416) pero no su nombre (L) (*).

Ejercicios

2.6. Escribir un programa que calcule el área del círculo utilizando instrucciones PRINT de tal manera que especifiquen las entradas de datos y los resultados.

(*) Algunos BASIC permiten escribir:

```
INPUT "INTRODUZCA EL RADIO"; R
```

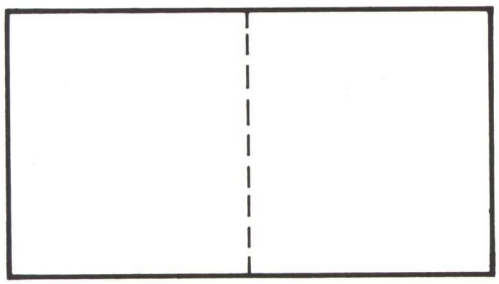
con los mismos efectos que los observados con la instrucción 5 PRINT "INTRODUZCA EL RADIO".

El «punto y coma» y la «coma» en BASIC

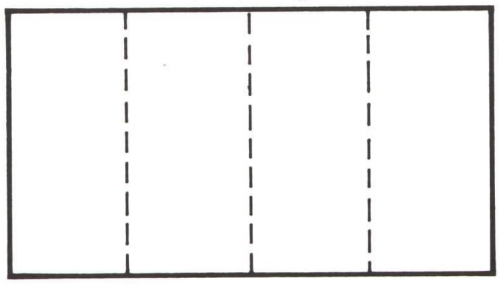
Un punto y coma (;) entre dos expresiones en una instrucción PRINT indica al microordenador que no debe dejar ningún espacio antes de imprimir el número o el texto que le sigue.

Una coma indica que la impresión siguiente se debe hacer en la siguiente zona de la pantalla.

Según el microordenador, la pantalla puede estar dividida en 2 ó 4 zonas.



Pantalla con dos zonas.



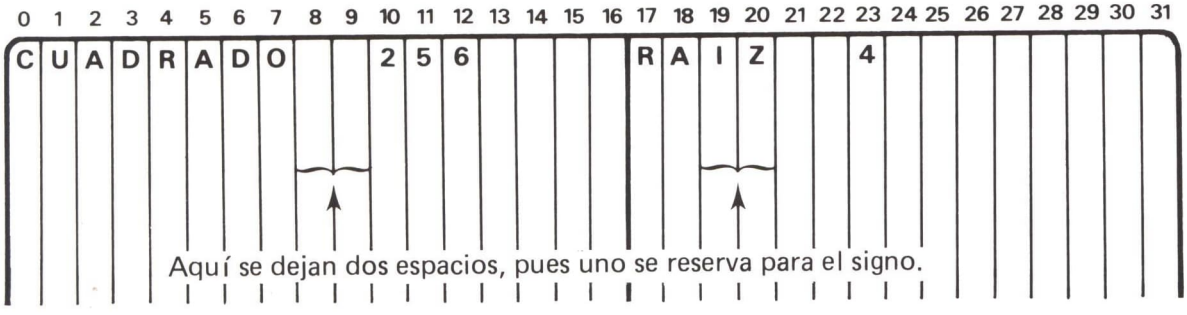
Pantalla con cuatro zonas.

Para comprender mejor la función de estos dos signos, veamos cómo es la ejecución del programa que calcula el cuadrado y la raíz cuadrada de un número N.

```
10  INPUT N
20  LET C=N ↑ 2
30  LET R=SQR (N)
40  PRINT ''CUADRADO:''; C, ''RAIZ:''; R
```

(*)

Si, por ejemplo, la pantalla tiene capacidad para 32 espacios y está dividida en dos zonas, para N=16, se obtendría el siguiente resultado en pantalla:



(*) La función SQR calcula la raíz cuadrada del número que se encuentra entre paréntesis.

Si el programa anterior lo hubiéramos escrito así:

```
10  INPUT N
20  LET C=N ↑ 2
30  LET R=SQR (N)
40  PRINT ''EL CUADRADO ES:''; C
50  PRINT ''LA RAIZ ES:''; R
```

después de teclear RUN y accionar [ENTER] se obtendría el siguiente resultado, para N=16.

```
EL CUADRADO ES: 256
LA RAIZ ES: 4
```

Es decir, si no hay coma o punto y coma al final de una sentencia PRINT, el siguiente PRINT se escribe en la línea siguiente.

Por otro lado, una instrucción en la que hubiera un PRINT sin nada detrás, haría que se dejara una línea en blanco entre el PRINT anterior y el siguiente. Así, si el programa fuera

```
10  INPUT N
20  LET C=N ↑ 2
30  LET R=SQR (N)
40  PRINT ''EL CUADRADO ES''; C
45  PRINT
50  PRINT ''LA RAIZ ES''; R
```

al ejecutarlo (RUN [ENTER]) aparecerá en pantalla, para N=16,

```
EL CUADRADO ES 256

LA RAIZ ES 4
```

Uso del PRINT para imprimir cálculos directamente

El programa anterior se podría simplificar más, de la siguiente manera:

```
10  INPUT N
20  PRINT N ↑ 2
30  PRINT SQR (N)
40  END
```

Al ejecutarlo (teclear RUN y accionar ENTER), para N=16, se obtendría este resultado en la pantalla pero sin almacenar previamente en C y R los resultados correspondientes a N ↑ 2 y SQR (N):

```
256
4
```

Es decir, una instrucción PRINT permite imprimir el resultado de una expresión aritmética escrita a continuación.

PRINT

Formato: PRINT { nombre de variables no entrecomillado / expresiones aritméticas / texto entre comillas }

Función: — escribe el valor de las variables;
— escribe el resultado de las expresiones aritméticas;
— escribe el texto entre comillas, pero no éstas.

Escritura de los números en lenguaje BASIC

Tecleamos el siguiente programa:

```
10  REM ESCRITURA DE NUMEROS
20  PRINT 7,-6328
30  PRINT 3.5,-34.270398
40  PRINT "POBLACION DE LA TIERRA=" ; 3600000000
50  END
```

Al ejecutarlo (teclear RUN y accionar ENTER) en la pantalla aparecen los siguientes resultados:

```
7                -6328
3,5              -34.270398
POBLACION DE LA TIERRA=3.6.E+9
```

corresponde a la línea 20
corresponde a la línea 30
corresponde a la línea 40

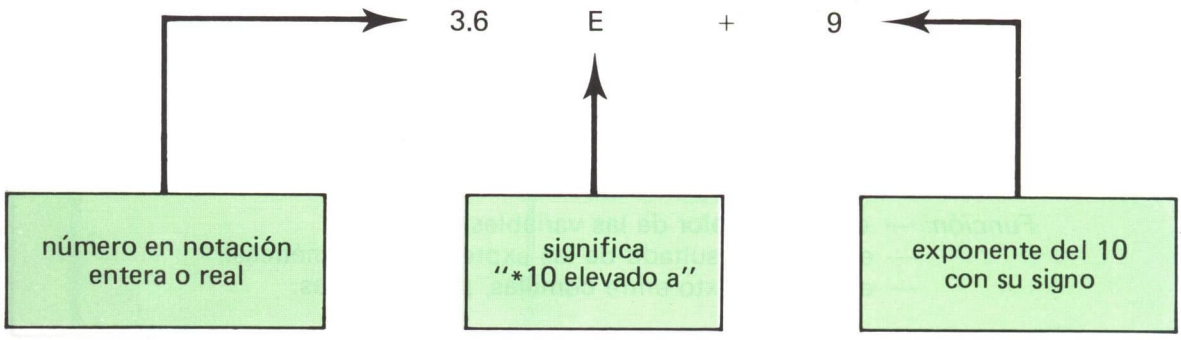
Los números que corresponden a la línea 20 no tienen punto decimal. Están escritos en notación *entera.*

Los números que corresponden a la línea 30 tienen punto decimal. Están escritos en notación *real.*

El número que corresponde a la línea 40 (población de la Tierra) está escrito en notación *exponencial.* El número en la forma 3600000000 se ha transformado en 3.6 E+9.

$$3600000000 = \underbrace{3.6E+9}_{\text{notación exponencial}}$$

La notación exponencial contiene tres partes:



¿Por qué el ordenador emplea esta notación? Porque en BASIC, normalmente, se pueden escribir números hasta con nueve cifras, un punto decimal y un signo (+ ó −). Si el número contiene más de nueve dígitos, automáticamente lo transforma a notación exponencial.

Esta notación es apta para escribir números muy grandes y muy pequeños, como, por ejemplo, la velocidad de un caracol en kilómetros por segundo:

```
10  REM VELOCIDAD DEL CARACOL
20  PRINT ''VEL. CARACOL=''; 0.000012
30  END
```

Después de teclear RUN y accionar [ENTER], se obtiene en pantalla

```
VEL. CARACOL=1.2 E-5
```

RESUMEN

En este capítulo se han visto las siguientes instrucciones:

Instrucción	Significado aproximado	Función
[ENTER] [RETURN]... o su equivalente	Entra, vuelve...	Hace que el dato o instrucción tecleada se almacene en memoria.
RUN	Corre	Inicia la ejecución de un programa.
NEW	Nuevo	Borra todo el contenido de la memoria.
LIST	Lista	Hace aparecer en pantalla un programa almacenado en la memoria.
REM	Comentario	Permite añadir aclaraciones a un programa sin afectarlo.
END	Fin	Señala el final de un programa.
INPUT	Introduce	Permite introducir datos desde el teclado.
LET	Sea	Asigna un valor a una variable en la que se almacena.
PRINT	Imprime	Imprime los resultados obtenidos en la pantalla.

EJERCICIOS DE RECAPITULACION

2.1. ¿Qué hay que hacer para que se ejecute el programa?

```
LET Y=3+4
PRINT Y
END
```

2.2. Supongamos que hemos detectado un cierto error en un programa. ¿Qué haríamos para corregirlo?

2.3. Si tenemos un programa en la pantalla (y en la memoria) de un microordenador, ¿cómo lo ejecutaríamos?

2.4. Escribir los símbolos empleados en lenguaje BASIC para las operaciones:
suma
resta
multiplicación
división
exponenciación

2.5. En los programas siguientes, ¿en qué orden se harían las operaciones?

a)
```
10 LET Y=8+2 * 5 ↑ 4
20 PRINT Y
30 END
```

b)
```
10 LET Y=9-24/35
20 PRINT Y
30 END
```

c)
```
10 LET Y=(8+3) * (5-2)
20 PRINT Y
30 END
```

2.6. ¿Qué aparecerá en pantalla al ejecutar los siguientes programas?

a)
```
10 PRINT 10, 20
20 END
```

b)
```
10 PRINT 10; 20
20 END
```

c)
```
10 PRINT 10
20 PRINT 20
30 END
```

d)
```
10 PRINT 10
20 PRINT
30 PRINT
40 PRINT 20
50 END
```

2.7. ¿Qué se imprime en pantalla al ejecutar el siguiente programa?

```
10  LET R=15
20  LET S=3.141592 * R ↑ 2
30  PRINT ''EL AREA DEL CIRCULO ES'', S
40  END
```

2.8. El siguiente programa contiene un error. ¿Cuál es?

```
10  INPUT P
20  INPUT Q
30  LET S=(P+Q)/R
40  PRINT S
50  END
```

2.9. Escribir un programa que dé el valor de un número cualquiera de duros en pesetas.

2.10. Escribir un programa que permita calcular el área S y el volumen V de una esfera de acuerdo con las fórmulas $S=4\pi R^2$ y $V=\frac{4}{3}\ \pi R^3$, introduciendo el valor del radio R desde el teclado.

2.11. Hacer un programa que transforme un ángulo en radianes; pasarlo a grados, minutos y segundos.

2.12. Elaborar un programa que calcule el número de variaciones con repetición de M elementos tomados de N en N.

2.13. Sabiendo que la suma de los N primeros términos de una progresión aritmética viene dada por la fórmula:

$$S=\frac{A1+AN}{2}\cdot N$$

siendo: A1 primer término de la sucesión,
AN término de posición N,
N número de términos,
elaborar un programa que lea A1, AN, N y dé como resultado la suma S.

2.14. La fórmula que transforma grados centígrados a grados Fahrenheit es:

$$F=\frac{9}{5}\cdot C+32$$

Hacer un programa que lea grados centígrados y escriba su equivalente en grados Fahrenheit.

2.15. Aplicando la fórmula del interés simple

$$I=\frac{C\cdot r\cdot t}{100}$$

hacer un programa que calcule el interés.

2.16. Dado un capital inicial de C pesetas a un interés I (tanto por uno), escribir un programa que permita calcular el capital acumulado C. F. al cabo de N años a interés compuesto, según la expresión

$$CF=C\ (1+I)^N$$

3 Saltos incondicionales y condicionales

Programa para escribir infinitas veces una frase

A veces es necesario que las instrucciones no se ejecuten secuencialmente, es decir, siguiendo los números. Para poder saltar instrucciones se introduce GOTO 10 que sirve para ir a la instrucción 10, es decir, para que el microordenador continúe su ejecución en la instrucción 10.

Vamos a escribir un programa que imprima infinitas veces la frase:

BURRO SE ESCRIBE CON B

El programa sería de la forma:

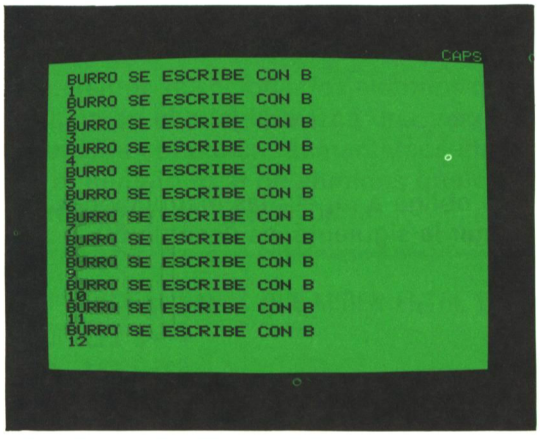
```
 1  REM ESTE PROGRAMA NO TIENE FIN
10  PRINT ''BURRO SE ESCRIBE CON B''
20  GOTO 10
30  END
```

Como dice el comentario, este programa no tiene fin, pues la instrucción 10 escribe la frase, la 20 devuelve el control a la 10, es decir, obliga al ordenador a saltar a la instrucción 10 necesariamente, que vuelve a escribir la frase, y así se repite el proceso. La instrucción 30 de fin no se ejecutará nunca.

En el programa anterior podemos introducir una variable C (contador) en la que se registra el número de veces que se repite la frase. El programa que resulta es:

```
 1  REM ESTE PROGRAMA NO TIENE FIN
 5  LET C=0
10  PRINT "BURRO SE ESCRIBE CON B"
15  LET C=C+1
17  PRINT C
20  GOTO 10
30  END
```

Al ejecutar el programa (RUN [ENTER]), en la pantalla aparecerán sucesivamente:



corresponde a la instrucción 10

corresponde a la instrucción 15

corresponde a la instrucción 10

corresponde a la instrucción 15, etc.

. . . . . .

. . . . . .

. . . . . .

En la instrucción 5 se ha introducido una variable C, cuyo valor inicial es cero. En la instrucción 15 se suma una unidad a C cada vez que se repite la frase. En la 17 se imprime el valor del contador C. Naturalmente, el número almacenado en el contador crece indefinidamente.

GOTO

Formato: GOTO n
Función: transfiere el control a la instrucción numerada con el número **n**.

Ejercicios

3.1. Elaborar un programa que escriba un número ilimitado de veces la frase

ME ESTOY MAREANDO DE ESCRIBIR SIEMPRE LA MISMA FRASE

e imprima el número de veces que la escribe.

3.2. Hacer un programa que escriba todos los números naturales.

3.3. Realizar un programa que escriba todos los números naturales impares.

Programa: obtener los múltiplos de 3 hasta un cierto número

Supongamos que nos dan un número N y queremos calcular e imprimir todos los múltiplos de 3 menores que N (se supone N mayor que tres).

El razonamiento lógico que debemos seguir es:

10 introducir N
20 M=3
30 escribir M
40 M=M+3
50 si M<N entonces ir a 30
60 fin

Aquí la instrucción importante es la 50, que obliga a repetir las instrucciones 30 y 40 siempre que M<N, y, en caso contrario, a ejecutar la siguiente, es decir, la 60.

El programa en BASIC es:

```
 1  REM MULTIPLOS DE TRES
10  INPUT N
20  LET M=3
30  PRINT M; ",";
40  LET M=M+3
50  IF M < N THEN 30
60  END
```

10 introducimos N (elegimos el valor 11)
20 M=3
30 Escribir M: 3, 6, 9 } lo que va escribiendo
40 M=M+3 (resultados sucesivos de M:6, 9)
50 Si M<N entonces ir a 30
Comparaciones sucesivas:
¿6<11? Sí; ir a 30, escribe 6
¿9<11? Sí; ir a 30, escribe 9
¿12<11? No; ir a 60
60 Fin

La instrucción 30 escribe cada valor de M seguido de una coma para ir separando los distintos números.

La 50 manda ir a la instrucción 30 si M<N y a la 60 en caso contrario.

La instrucción 60 END indica el fin del programa.

Introducimos el programa pulsando a continuación de cada instrucción la tecla ENTER .

Intentamos ejecutarlo:

RUN [ENTER]

Aparece en pantalla:

?

Le damos el dato N=29:

29 [ENTER]

y aparece el resultado:

3, 6, 9, 12, 15, 18, 21, 24, 27

Ejercicios

3.4. Elaborar un programa que obtenga todos los múltiplos de 7 hasta un cierto número.

3.5. Hacer un programa que dé primero todos los múltiplos de 5 y después todos los múltiplos de 11 hasta un cierto número.

Programa: recuento de alumnos que miden entre 1.60 y 1.70

Si queremos seleccionar los alumnos que miden entre 1.60 y 1.70 tenemos que escribir un programa en el cual aparezca una instrucción condicional IF... THEN..., de tal forma que si se cumple la condición, aumentamos en una unidad un contador, para, posteriormente, imprimir el número total de alumnos que cumplen la condición.

El programa sería de la forma:

```
 1  REM ALUMNOS QUE MIDEN ENTRE 1.60 y 1.70
10  LET S=-1
20  LET S=S+1
30  PRINT "EL NUMERO TOTAL, HASTA EL MOMENTO ES: ", S
40  INPUT X
50  IF X>1.60 AND X<1.70 THEN 20                    (*)
60  GOTO 30
70  END
```

En la instrucción 10 iniciamos la cuenta con S=−1 para que en la 20 se haga S=0, con lo que nuestro contador se pone a cero antes de introducir un valor por medio de la instrucción 40.

La instrucción fundamental es la:

```
50 IF X>1.60 AND X<1.70 THEN 20
```

ya que si la altura X de un alumno determinado es mayor que 1.60 y (AND) además X es menor que 1.70, entonces el control se transfiere a la instrucción 20 para que aumente en una unidad el contador S y siga el proceso con otros alumnos.

(*) En algunos microordenadores es necesario poner THEN GOTO 20.

En caso contrario (si el alumno no mide entre 1.60 y 1.70) se ejecuta la instrucción 60 GOTO 30 que nos envía el control a la 30 sin pasar por la 20 y, por tanto, no aumentando el contador S.

Comprobación del programa (ejecución)

Introducimos el programa tecleando cada instrucción y el comando [ENTER] al terminar de escribir cada una de ellas.

Para ejecutarlo tecleamos

```
RUN [ENTER]
```

Aparece en pantalla

```
EL NUMERO TOTAL HASTA EL MOMENTO ES: 0
?
```

Damos el primer valor, por ejemplo,

```
1.63
```

Como 1.63 está entre 1.60 y 1.70, nos aparece en pantalla:

```
EL NUMERO TOTAL HASTA EL MOMENTO ES: 1
?
```

Introducimos un segundo valor, por ejemplo,

```
1.85
```

Pero 1.85 no se encuentra entre 1.60 y 1.70, luego el número no varía:

```
EL NUMERO TOTAL HASTA EL MOMENTO ES: 1
?
```

Así seguiríamos dando valores hasta contabilizar el número que nos interese.

IF-THEN

Formato: IF E operador de relación E' THEN n.

Función: evalúa la expresión E y la E' y las relaciona mediante el operador. Si la relación se cumple, salta a la instrucción numerada con **n**; en caso contrario, sigue ejecutando la instrucción siguiente a la IF THEN.

Entre IF y THEN pueden intercalarse más de una relación del tipo *E operador de relación E',* vinculadas mediante los operadores lógicos: AND, OR y NOT.

RELACIONES

Las relaciones que normalmente se suelen utilizar son las siguientes:

A< B	A es menor que B.
A <=B	A es menor o igual que B.
A<>B	A es distinto de B.
A>=B	A es mayor o igual que B.
A>B	A es mayor que B.
A=B	A es igual a B.
P AND Q	Se verifica la relación P y la Q.
P OR Q	Se verifica la relación P o la Q.
NOT P	No se verifica la relación P.

A y B son expresiones aritméticas.

Ejercicios

3.6. Hacer un programa que dé el número de alumnos cuyo peso está comprendido entre 50 y 60 kilos.

3.7. Elaborar un programa que calcule el número de alumnos cuyo peso es mayor que 80 kilos y su altura inferior a 1.80 metros.

3.8. Elaborar un programa que dé los múltiplos comunes de 3 y 7 hasta un cierto número N.

Programa: calcular valores medios

Una aplicación frecuente de la instrucción IF-THEN es la utilización de un indicador para terminar un proceso.

Analicemos este programa

```
 10  REM CALCULA EL VALOR MEDIO
 20  PRINT "VALOR MEDIO"
 30  PRINT "INTRODUZCA LOS NUMEROS"
 40  PRINT "CUANDO TERMINE ESCRIBA-1"
 50  PRINT
 60  LET T=0
 70  LET N=0
 80  INPUT X
 90  IF X=-1 THEN 130
100  LET T=T+X
110  LET N=N+1
120  GOTO 80
130  PRINT
140  PRINT "EL NUMERO DE VALORES ES: " ; N
150  PRINT "EL VALOR MEDIO ES: " ; T/N
160  END
```

Al ejecutar el programa (RUN ENTER), si introducimos los valores

X=8
X=7
X=5
X=4
X=-1

imprimirá en la pantalla

```
EL NUMERO DE VALORES ES: 4
EL VALOR MEDIO ES: 6
```

En 60 y 70 definimos las variables T y N que sirven para ir almacenando las sumas y el número de valores introducidos, respectivamente.

En 90, si el número introducido es −1, se termina la introducción de valores (si no, seguiría indefinidamente). Si X=−1, se ejecutan las instrucciones 130 a 160 que después de dejar una línea en blanco imprimen el número de valores (instrucción 140) y el valor medio (instrucción 150).

Para parar la ejecución de un programa: la instrucción STOP

Utilizando esta instrucción se puede *detener* la ejecución de un programa en cualquiera de sus líneas.

Así, en el programa:

```
 1  REM NOTAS
10  PRINT ''INTRODUZCA UNA NOTA''
20  INPUT N
30  IF N>5 THEN 60
40  PRINT ''SUSPENSO''
50  STOP
60  PRINT ''APROBADO''
70  END
```

la instrucción

```
50  STOP
```

si N es menor que 5, detiene la ejecución del programa después de aparecer en la pantalla la palabra SUSPENSO. Es decir, la ejecución se detiene en la línea 50. La instrucción STOP para la ejecución del programa en la línea en que se encuentra, pero, a diferencia de la instrucción END que siempre debe estar situada al final, puede aparecer en cualquier línea y las veces que sea necesario.

El efecto que produce la instrucción

```
50  STOP
```

es, por otra parte, el mismo que hubiera producido la instrucción

```
50  GOTO 70
```

STOP

Formato: STOP.
Función: detiene la ejecución del programa justo en la línea en que se encuentra STOP.

RESUMEN

Instrucción	Significado aproximado	Descripción
GOTO	Ir a	Rompe la ejecución del programa según la secuencia numérica consecutiva para desplazar el control a otra instrucción anterior o posterior.
IF...THEN n. (en algunos microordenadores esta instrucción tiene la forma IF THEN GOTO n).	Si... entonces	Si se cumple la relación escrita entre IF y THEN, se transfiere el control a la instrucción n. En caso contrario, se ejecuta la instrucción siguiente a la IF-THEN.
STOP	Parada	Detiene la ejecución de un programa.

EJERCICIOS DE RECAPITULACION

3.1. ¿Qué se imprimirá en pantalla cuando se ejecute el siguiente programa?

```
10  LET R=15
20  GOTO 99
30  PRINT R
99  END
```

3.2. ¿Qué imprimirá el siguiente programa para X=4? ¿Y para X=83?

```
10  INPUT X
20  IF X< 50 THEN 40
30  STOP
40  PRINT X
50  GOTO 10
60  END
```

3.3. Escribir las siguientes condiciones como instrucciones IF-THEN.

a) Si el valor de X es menor o igual que cero pase a la instrucción 100.
b) Si el valor de B1 es mayor o igual que el valor de B2 pase a la instrucción 350.
c) Si el valor de X es mayor que Y y mayor que Z pase a la instrucción 500.
d) Si el valor de X es mayor que Y o mayor que Z pase a la instrucción 320.

3.4. Escribir las siguientes condiciones:

a) Si el valor de Z es mayor o igual que el valor de A al cuadrado entonces pase a la línea 150.
b) Si el valor de 3 veces A es igual a 12 entonces pase a la línea 80.

3.5. Escribir una o varias instrucciones necesarias para que:

a) Si X tiene un valor mayor que Y+2 se transfiera el control a la instrucción 200; en caso contrario ir a 120.
b) Si J tiene un valor cero se transfiera el control a 150, y si no es así aumente el valor de J en una unidad.

3.6. ¿Cuál de estas instrucciones es correcta? ¿Por qué?

a) IF X ↑ 2=100 THEN 50
b) IF X1< =50 THEN X=X+5

3.7. ¿Qué imprimirá el programa...

```
 5  LET T=0
10  INPUT X
20  LET T=T+5 * X
30  PRINT T
40  GOTO 10
```

para los datos X=8, 7, 5? ¿Qué instrucción o instrucciones se pueden añadir para que la ejecución del programa termine después de que se imprima T para estos valores de X?

3.8. Indíquese los resultados que aparecen en pantalla si se ejecuta el siguiente programa

```
10  INPUT X
20  IF X<0 THEN 10
30  PRINT X
40  GOTO 10
```

para X=5, −3, 6, 0, 8, −1.

3.9. Si se ejecuta el siguiente programa, que pretende clasificar dos números A y B,

```
10  INPUT A, B
20  IF A<B THEN 50
30  IF A>B THEN 60
40  IF A=B THEN 70
50  PRINT A; "ES MENOR QUE"; B
60  PRINT A; "ES MAYOR QUE"; B
70  PRINT A; "ES IGUAL QUE"; B
80  GOTO 10
90  END
```

al introducir dos números, por ejemplo, 5 y 7, obtenemos:

```
5 ES MENOR QUE 7
5 ES MAYOR QUE 7
5 ES IGUAL QUE 7
```

¿Cómo se puede corregir este fallo?

3.10. Hacer un programa en el que se lean los valores A y B, y A > B entonces dé como resultado A−B, y en caso contrario dé B−A.

3.11. Elaborar un programa que lea dos números y, si el primero es mayor que el segundo, escriba su diferencia y su cociente; en caso contrario, su suma y su producto.

3.12. Se hace una encuesta preguntando a los ciudadanos sobre un determinado producto. Las respuestas favorables al producto se consideran 1 y las desfavorables 0. Construir un programa que permita contar el número de respuestas favorables y desfavorables a dicho producto.

3.13. Escribir un programa que permita calcular el número de años en que se duplica una cierta cantidad de dinero a interés compuesto.
(Si P es el capital, R el interés, Y los años y P1 el capital en un año, los intereses son I=P1 * R/100 y el capital acumulado en ese año P1=P1+I).

3.14. Escribir un programa que permita calcular la velocidad alcanzada en un movimiento uniformemente acelerado ($v=v_o+at$) siendo v_o la velocidad inicial, v la final, t el tiempo y a la aceleración. Contemplar el caso en el que la aceleración sea negativa, pudiendo, en consecuencia, llegar a anularse la velocidad.

3.15. Dado un conjunto de números en orden ascendente, escribir un programa que calcule la media. El último valor viene señalado por −1000.

3.16. Elaborar un programa que lea un número N y dé un informe si es un número primo o no.

3.17. Dados los tres lados de un triángulo, hacer un programa que indique si es isósceles, equilátero o escaleno.

3.18. Realizar un programa que clasifique un triángulo en rectángulo, obtusángulo o acutángulo, dados sus tres ángulos.

3.19. Elaborar un programa que introduzca dos ángulos en grados, minutos y segundos; sume y escriba el resultado en grados, minutos y segundos, de tal forma que los minutos sean inferiores a 60 y los segundos también.

3.20. Hacer un programa que reconozca si el número introducido es par o impar, dando el informe correspondiente.

3.21. Hacer un programa que introduzca números sucesivamente y dé un informe con el mayor introducido hasta el momento.

3.22. Rectificar el programa anterior para que vaya poniendo en una primera columna el número que se ha introducido y en una segunda columna el mayor número introducido hasta el momento. Además, como cabeceras de ambas columnas, poner

NÚMERO EL MAYOR

3.23. Dado un ángulo en grados, minutos y segundos, hacer un programa para pasarlo a radianes.

3.24. Para calcular el máximo común divisor de dos números se puede aplicar el teorema:

si X>Y entonces M.C.D. (X, Y)=M.C.D. (X−Y, Y)

Por ejemplo:

M.C.D. (80, 30)=M.C.D. (50, 30)=M.C.D. (20, 30)
=M.C.D. (20, 10)=M.C.D. (10, 10)
=10

Realizar un programa que dé el máximo común divisor de dos números, aplicando el teorema anterior.

3.25. Dados cuatro números A, B, C y D, escribir un programa que haga I=1 si los cuatro son iguales, I=2 si son tres iguales, I=3 si son dos iguales, I=4 si los números son iguales dos a dos, e I=5 si todos son diferentes.

4 Datos de un programa

Instrucciones READ y DATA

Cuando en un programa se va a trabajar con un volumen relativamente grande de datos, las instrucciones LET e INPUT resultan lentas e incómodas. Se utilizan entonces las instrucciones READ y DATA, las cuales siempre trabajan juntas.

Las instrucciones DATA son simples almacenes de datos separados por *comas*. Por ejemplo:

```
10 DATA 1.5, 2.5, 3.1, 11, 0.6
20  DATA -3.5, 4
```

En un programa pueden ir tantas como se quiera y situadas en cualquier línea, aunque por motivos de orden y claridad, conviene que estén todas juntas, bien al principio o bien al final del programa.

Cuando el ordenador encuentra una sentencia DATA no la ejecuta. Pasa por ella hasta que encuentre una sentencia READ.

La instrucción READ *manda al ordenador que lea* los datos que se encuentran en la instrucción DATA correspondiente y que no han sido leídos todavía. Es como si la máquina cogiera un «índice» y fuera recorriendo los datos contenidos en DATA hasta terminar de leer. En este momento el «índice» se para y los datos que sobran quedan como en una «lista de espera» preparados para ser leídos cuando se encuentre una nueva instrucción READ. Así, si las instrucciones

```
30  READ A, B, C, D, E
40  READ F, G
```

formaran con las 10 y 20 parte de un único programa real, al ejecutarse el programa, en la memoria se almacenarían estos valores:

A = 1.5
B = 2.5
C = 3.1
D = 11
E = 0.6
F = −3.5
G = 4

No importa que algunos datos se queden sin leer. Pero, por el contrario, si se intentan leer más datos de los disponibles, aparece en la pantalla un mensaje de error y el programa se interrumpe.

El mensaje suele ser ? OUT OF DATA ERROR IN...

Ejemplo: Programa que calcula e imprime en pantalla el cuadrado de los nueve primeros números naturales:

```
10 READ N
20 LET C=N * N
30 PRINT "EL CUADRADO DE" ; N ; "ES" ; C
40 GOTO 10
50 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
100 END
```

La instrucción 40 GOTO 10 haría ejecutar el programa indefinidamente intentando leer infinitos datos. Pero como la instrucción 50 DATA tiene un número limitado de datos, al agotarse éstos da un mensaje de error, despues de imprimir el cuadrado de los números.

Si, además, quisiéramos obtener los cuadrados de los números naturales del 9 al 15 añadiríamos la instrucción

```
60 DATA 10, 11, 12, 13, 14, 15
```

READ y DATA

Formato: READ lista de variables.
DATA lista de constantes.
Función: se asigna a las variables que figuran en READ las constantes que aparecen en DATA, secuencialmente.

Instrucción RESTORE

Si en un programa los datos almacenados van a ser necesarios otra vez, después de haber sido leídos, es preciso que no sean borrados al pasar por ellos el «índice».

La instrucción RESTORE regenera los valores de DATA y hace que el «índice» vuelva al principio de la lista para releer de nuevo los mismos datos al ejecutar una nueva instrucción READ, siempre dentro del mismo programa.

Así, el programa

```
10 READ A, B, C
20 PRINT A; B; C
30 READ A, B, C
40 PRINT A; B; C
50 RESTORE
60 READ A, B, C
70 PRINT A; B; C
80 DATA 1, 2, 3, 4, 5, 6
90 END
```

imprime en pantalla:

```
1 2 3
4 5 6
1 2 3
```

Si se quita la instrucción

```
50 RESTORE
```

la ejecución del programa se detiene en 60 y aparece un mensaje de error.

RESTORE

Formato: RESTORE.
Función: permite la repetición de la lectura o asignación de un grupo de datos asociados a instrucciones READ y DATA.

RESUMEN

Instrucción	Significado aproximado	Función
READ A, B	Leer	Atribuye a las variables los valores colocados en un DATA.
DATA 4, 7, 18	Datos	Contiene los datos que serán leídos en una instrucción READ.
RESTORE	Restaura	Hace posible volver a leer una lista de datos contenida en una instrucción DATA.

EJERCICIOS DE RECAPITULACION

4.1. Las siguientes instrucciones contienen errores. ¿Cuáles son?

a) 10 READ X;Y
b) 90 DATA 5.5, 50, 7.5, 100

4.2. Un programa en BASIC contiene las instrucciones

```
30 READ A, B, C
.....
60 RESTORE
70 READ W, X, Y, Z
.....
200 DATA 1, 3, 5, 7, 9, 11, 13
```

¿Qué valores se asignan a cada variable? ¿Qué valores se les asignarían si no estuviese la instrucción 60 RESTORE?

4.3. Escribir las instrucciones READ y DATA que asignan los valores −1.6 E −6, −500, 0.4077, 100, 110 a las variables C1, C2, C3, C4, C5.

4.4. ¿Qué imprimiría en pantalla el microordenador al ejecutar este programa?

```
10 READ A, B
20 PRINT A, B
30 READ C
40 PRINT S=A+B+C
50 DATA 2, 7
60 END
```

4.5. En un observatorio se han registrado en una semana las siguientes temperaturas:

D	L	M	X	J	V	S
20	22	25	24	23	21	20

Escribir un programa que calcule la media, introduciendo los datos con las instrucciones READ - DATA.

4.6. En una encuesta se obtuvieron 50 respuestas. Cada respuesta vale 1 (sí) ó 2 (no). La última respuesta está seguida del valor −1. Escribir un programa que incluya las siguientes instrucciones DATA y que cuente el número de cada respuesta.

```
900 DATA 1, 2, 2, 1, 2, 2, 1, 1, 2, 1
910 DATA 2, 1, 1, 1, 2, 1, 2, 2, 2, 1
920 DATA 1, 2, 1, 2, 1, 1, 2, 1, 2, 2
930 DATA 1, 1, 1, 2, 2, 1, 1, 2, 2, 2
940 DATA 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, -1
```

4.7. Dado el programa:

```
  1 REM IMPRIME UN DIBUJO
 10 READ X
 20 IF X=-1 THEN 920
 30 IF X=1 THEN 80
 40 IF X=2 THEN 100
 50 IF X=3 THEN 120
 60 IF X=4 THEN 140
 65 IF X=5 THEN 160
 70 GOTO 10
 80 PRINT "*******     ***     *******"
 90 GOTO 10
100 PRINT "***    ****       *****  ***"
110 GOTO 10
120 PRINT "*  *  *  *  *  *  *  *  *  *  *"
130 GOTO 10
140 PRINT "* * * * * * * * * * * * * * * * * * * *"
150 GOTO 10
160 PRINT "* * *   * * *   * * *   * * *"
170 GOTO 10
900 DATA 3, 4, 4, 5, 1, 5, 1, 5, 1, 5, 2, 2
910 DATA 2, 5, 1, 5, 1, 5, 1, 5, 1, 5, 4, 4, 3, -1
920 END
```

¿Qué dibujo imprime? Hacer programas que realicen dibujos distintos.

5 Bucles

Bucles en un programa

Una de las principales cualidades de los ordenadores es la de poder ejecutar un gran número de procesos en un tiempo muy corto. Esta cualidad se aprovecha en los denominados *bucles,* donde el ordenador recorre un mismo grupo de instrucciones (bucle) repetidas veces.

Consideremos el programa siguiente, que permite calcular el cuadrado de los números comprendidos entre 1 y 5:

```
 1  REM CUADRADOS
10  FOR I=1 TO 5
20  LET A=I ↑ 2                                    (*)
30  PRINT ''EL CUADRADO DE'' ; I ; ''ES'' ; A
40  NEXT I
50  END
```

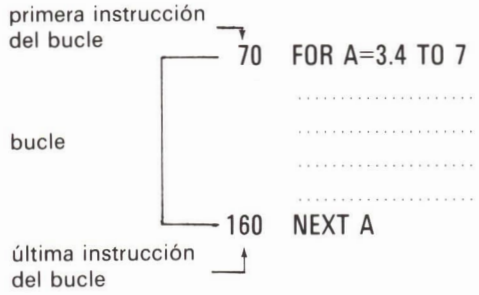
Una vez escrito el programa, al ejecutarlo (RUN ENTER) veremos en la pantalla:

```
EL CUADRADO DE 1 ES 1
EL CUADRADO DE 2 ES 4
EL CUADRADO DE 3 ES 9
EL CUADRADO DE 4 ES 16
EL CUADRADO DE 5 ES 25
```

La instrucción 10 FOR I=1 TO 5 hace que para I=1, para I=2, para I=3, para I=4 y para I=5 se ejecuten las instrucciones 20 y 30. La instrucción 40 NEXT I (siguiente I) es la que hace ir tomando el valor de I siguiente al considerado la vez anterior. Cuando I alcanza el valor 5, ejecuta por última vez el bucle, transfiriendo el control a la instrucción 50.

(*) En lugar de I ↑ 2 se puede escribir I * I.

Observamos que utilizando las dos instrucciones FOR-NEXT podemos ejecutar repetidamente un conjunto de instrucciones, denominado *ciclo* o *bucle*. La instrucción FOR es la primera del ciclo y NEXT, la última.



El programa anterior se podría modificar en el caso de que se desee calcular el cuadrado de los números comprendidos entre 1 y N, siendo N un número cualquiera.

```
 1  REM CUADRADOS
 5  INPUT N
10  FOR I=1 TO N
20  LET A=I ↑ 2
30  PRINT "EL CUADRADO DE" ; I ; "ES" ; A
40  NEXT I
50  END
```

La ejecución de este programa se interrumpe inmediatamente, apareciendo una ? o una L, según el microordenador. Para que continúe la ejecución se introduce desde el teclado un valor de N. Una vez introducido, sigue la ejecución hasta que I alcance el valor N inclusive, y una vez ejecutado el bucle para I=N se transfiere el control a la instrucción siguiente a la 40 NEXT I, es decir, a la 50 END.

Para que un bucle funcione es necesario asignar a la variable que se encuentra después de la palabra FOR, un valor *inicial,* y después de la palabra TO, un valor *final* o *tope.* Si no se añade nada más, cada vez que se ejecutan todas las instrucciones comprendidas entre FOR y NEXT se incrementa en 1 el valor de la variable, y así sucesivamente hasta que ésta alcanza un valor igual al tope; si es igual, se ejecuta el bucle por última vez. Así, en la instrucción

la variable se incrementa de 1 en 1.

Ejercicios

5.1. Hacer un programa que imprima la raíz cuadrada de los diez primeros números naturales.

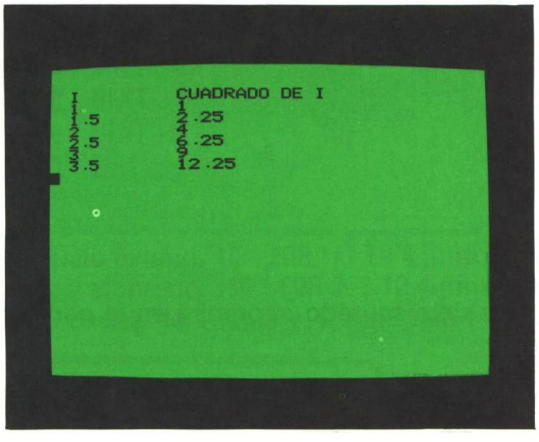
5.2. Esribir un programa que imprima veinte veces la frase "NO POR MUCHO CORRER SE LLEGA ANTES".

Bucles con incremento distinto a 1

En su forma más simple, la instrucción FOR.. TO..., avanza incrementando en 1 la variable de control hasta llegar al valor final de la variable. Sin embargo, el incremento puede hacerse diferente a 1 añadiendo la palabra STEP, como se ve en el siguiente programa que calcula los cuadrados de los números comprendidos entre 1 y 3.5 con incrementos de 0.5 y los coloca en forma de tabla.

```
 1  REM CUADRADOS
10  PRINT "I", "CUADRADO DE I"
20  FOR I=1.0 TO 3.5 STEP 0.5
30  LET A=I ↑ 2
40  PRINT I , A
50  NEXT I
60  END
```

Al ejecutar el programa (RUN [ENTER]) aparece en pantalla:



Observamos en la instrucción 20 FOR I=1.0 TO 3.5 STEP 0.5 que los valores iniciales y finales no tienen por qué ser necesariamente enteros y que la variable I se incrementa también en un valor no entero, con la utilización de la palabra STEP.

El incremento de la variable inicial también puede ser negativo, como se observa en el siguiente programa

```
 1  REM CUADRADOS
10  PRINT "I", "CUADRADO DE I"
20  FOR I=3.5 TO 1 STEP-0.5
30  LET A=I ↑ 2
40  PRINT I, A
50  NEXT I
60  END
```

que imprime la tabla anterior invertida

```
 1  REM CUADRADOS
10  PRINT "I" , "CUADRADO DE I"
20  FOR I=3.5 TO 1 STEP-0.5
30  LET A=I ↑ 2
40  PRINT I, A
50  NEXT I
60  END
```

RUN

ENTER

I	CUADRADO DE I
3.5	12.25
3	9
2.5	6.25
2	4
1.5	2.25
1	1

Ejercicios

5.3. Diseñar un programa que imprima los 25 primeros números pares.

5.4. Sabiendo que el primer término de una progresión aritmética es A=15 y la diferencia D=−2, calcular, por medio de un bucle, todos los términos de la progresión hasta el de posición 30.

5.5. Por medio de un bucle hacer una lista con los treinta primeros números impares negativos.

También pueden ponerse en la inicialización de un bucle expresiones aritméticas. Así, el programa

```
10  LET Q=4
20  FOR P=Q TO 2 * Q-1
30  PRINT P
40  NEXT P
50  END
```

permite averiguar la extensión del bucle. Después de ejecutado imprimirá en la pantalla

```
4
5
6
7
```

En forma análoga, este programa, una vez ejecutado

```
10  LET Q=4
20  FOR P=Q/2 TO Q * 3
30  PRINT P ;
40  NEXT P
50  END
```

haría que apareciesen en la pantalla los números

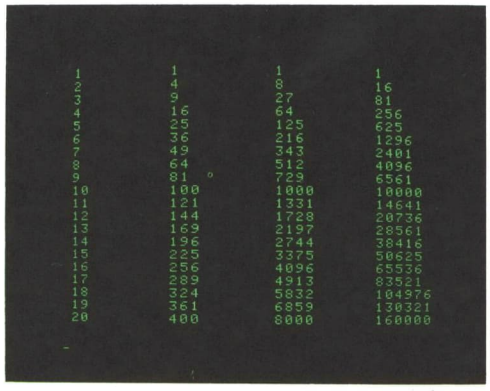
```
2  3  4  5  6  7  8  9  10  11  12
```

Bucles en el interior de otros bucles (bucles anidados)

Dentro de un bucle puede haber cualquier conjunto de instrucciones, e incluso otros bucles, como se ve en el programa siguiente que calcula las potencias segunda, tercera y cuarta de un número comprendido entre 1 y 4, ambos inclusive.

```
 1  REM POTENCIAS
10  FOR I=1 TO 4
20  PRINT I,
30  FOR J=2 TO 4
40  LET A=I ↑ J
50  PRINT A,
60  NEXT J
70  NEXT I
80  END
```

RUN → ENTER



El bucle exterior 10 FOR I=1 TO 4 junto con 70 NEXT I proporciona los números enteros de 1 a 4, y el interior (30 FOR J=2 TO 4 junto con 60 NEXT J) calcula las potencias segunda, tercera y cuarta de 1, 2, 3 y 4. Así, para I=1 y J=2, J=3 y J=4 imprime:

```
1       1       1       1
```

Para I=2 y J=2, J=3 y J=4 imprime:

```
2       4       8       16
```

y así sucesivamente.

Es importante advertir que un bucle interno tiene que estar incluido en su totalidad en otro más externo.

---

* Se ha colocado una coma (,) después de A en la instrucción 50 para que los valores de las potencias de los cuatro números aparezcan repartidos en las cuatro zonas de la pantalla. En esta ocasión, se ha supuesto que la pantalla tiene cuatro zonas.

```
10  FOR I=.....
20  FOR J=.....
30  FOR K=.....
    .....
    .....
80  NEXT K
90  NEXT J
100 NEXT I
```

La disposición de los tres bucles en el siguiente programa es incorrecta:

```
10  FOR I=.....
20  FOR J=.....
30  FOR K=.....
    .....
    .....
80  NEXT I
90  NEXT J
100 NEXT K
```

Ejercicios

5.6. Generar la tabla de multiplicar utilizando dos bucles.

5.7. Imprimir los resultados posibles que se obtienen al considerar la suma de los valores del lanzamiento de dos dados.

FOR... TO... STEP y NEXT

Formato: FOR V=i TO t STEP s
NEXT V

Función: Ejecuta un ciclo o bucle que empieza en FOR y termina en NEXT, iniciándose el ciclo cuando la variable V toma el valor inicial **i** y terminando después de alcanzar el valor tope o final **t**. El valor de la variable V incrementa su valor en función del incremento **s**.

RESUMEN

Instrucción	Significado aproximado	Función
FOR... TO... STEP...	Desde... a... en pasos de...	Ejecuta el conjunto de instrucciones comprendidas entre FOR y NEXT un número determinado de veces (bucle).
NEXT	Siguiente	Transfiere el control a la instrucción FOR-TO si el valor de la variable es menor o igual que el tope.

EJERCICIOS DE RECAPITULACION

5.1. Sean los dos programas siguientes:

```
10  LET F=1
20  IF F>8 THEN 99
30  PRINT "F= "; F
40  LET F=F+1
50  GOTO 20
99  END
```

```
10  FOR F=1 TO 8
20  PRINT "F= "; F
30  NEXT F
99  END
```

¿Qué imprimirá cada uno de ellos en pantalla?

5.2. ¿Qué imprimirá en pantalla el siguiente programa?

```
10  LET X=0
20  FOR K=1 TO 4
30  LET X=X+K
40  NEXT K
50  PRINT X
60  END
```

5.3. ¿Qué imprimirá en pantalla el siguiente programa?

```
10  LET P=1
20  FOR K=1 TO 4
30  LET P=P*K
40  NEXT K
50  PRINT P
60  END
```

5.4. Dado el programa

```
10  LET N=1
20  FOR K=1 TO N
30  PRINT " * "
40  NEXT K
50  PRINT
60  LET N=N+1
70  IF N >10 THEN 99
80  GOTO 20
90  END
```

¿Qué imprimirá en pantalla al ejecutarlo?

5.5. ¿Qué imprimirá el siguiente programa?

```
10  LET S=0
20  FOR K=1 TO 7 STEP 2
30  LET S=S+K
40  NEXT K
50  PRINT S
60  END
```

5.6. ¿Son correctos los programas que incluyen las siguientes instrucciones?

a)
```
20 FOR I=1 TO 100 STEP I
.....
80 NEXT J
```

b)
```
50 FOR N=N1 TO N2
.....
.....
90  NEXT N
.....
120 IF N=10 THEN 75
.....
```

5.7. ¿Son correctos los programas que incluyen las siguientes instrucciones?

a)
```
100 FOR K=3 TO -3 STEP -1
.....
130 PRINT X ↑ K
.....
150 NEXT K
```

b)
```
25 FOR X=1 TO 2 STEP 0.05
.....
50 FOR Y=1 TO 10 STEP 0.1
.....
75 NEXT X
.....
100 NEXT I
.....
```

c)

```
100 FOR I=1 TO M
  .....
125 FOR J=1 TO N
  .....
135 FOR K=1 TO M+N
  .....
160 NEXT K
  .....
180 NEXT J
  .....
200 FOR J=1 TO N
  .....
225 NEXT J
235 FOR K=1 TO M+N
  .....
240 NEXT K
  .....
250 NEXT I
```

5.8. Escribir un programa que calcule la media de una lista de N números utilizando un INPUT para introducir N y después un ciclo FOR-NEXT en el que esté incluido un INPUT o un READ para introducir cada número.

5.9. Escribir un programa que permita calcular e imprimir la suma de los números enteros de 1 a N siendo N un valor introducido por una instrucción INPUT.

5.10. Escribir un programa para imprimir una tabla de las raíces quintas de los números comprendidos entre 1 y 2 en intervalos de 0.1

5.11. Escribir un programa para calcular la suma de los primeros cincuenta términos de una progresión geométrica de primer término F y razón R. (La suma es $F+F*R+F*R^2+\text{—}+F*R^{49}$.)

5.12. La sucesión de Fibonacci comienza por dos unos y los siguientes se obtienen sumando los dos anteriores.
1, 1, 2, 3, 5, 8...
Escribir un programa que lea un número N y genere los N primeros términos de la sucesión de Fibonacci.

5.13. Elaborar un programa que calcule la suma: $1^2+2^2+3^2+...+99^2$.

5.14. Hacer un programa que calcule los N primeros números primos.

5.15. Hacer un programa que calcule el producto de los N primeros números naturales (sin contar el cero), es decir, el factorial de N (N!).

5.16. Elaborar un programa que calcule el número de variaciones de M elementos tomados de N en N.

5.17. Elaborar un programa que calcule el número de combinaciones de M elementos tomados de N en N.

6 Cadenas

Cadenas de caracteres

Se llama *cadena de caracteres* a una sucesión de signos gráficos cualesquiera, letras, números y demás símbolos que se escriben entre comillas para saber el comienzo y el final de la cadena.

En las cadenas, los espacios en blanco se consideran también como caracteres.

Ejemplos de cadenas:

```
"A35, 8!"
"374" (que no es lo mismo que el número 374)
"MARIA"
"AIB2L3"
"JUAN PEDRO"
```

Hemos visto que en BASIC los números se «guardan» en variables. De la misma manera, las cadenas se guardan en *variables de cadena* también llamadas *variables alfanuméricas;* se distinguen de las numéricas porque su nombre siempre debe terminar con el símbolo $ (símbolo de dólar).

Ejemplos de variables alfanuméricas:

```
A$
BI$
FF5A$
```

(En algunos microordenadores sólo se puede poner una letra seguida de $.)

Ejemplo:

```
 1 REM ESCRITOR DE FRASES
10 INPUT A$
20 PRINT "LA FRASE ES: " ; A$
30 END
```

Si al ejecutar el programa, después de aparecer ? tecleamos «MI NOMBRE ES JUAN», aparecerá en pantalla:

```
MI NOMBRE ES JUAN
LA FRASE ES: MI NOMBRE ES JUAN
```

Ejercicios

6.1. Hacer un programa que dé los nombres y apellidos de una persona.

6.2. Hacer un programa que dé el nombre y la edad de una persona infinitas veces.

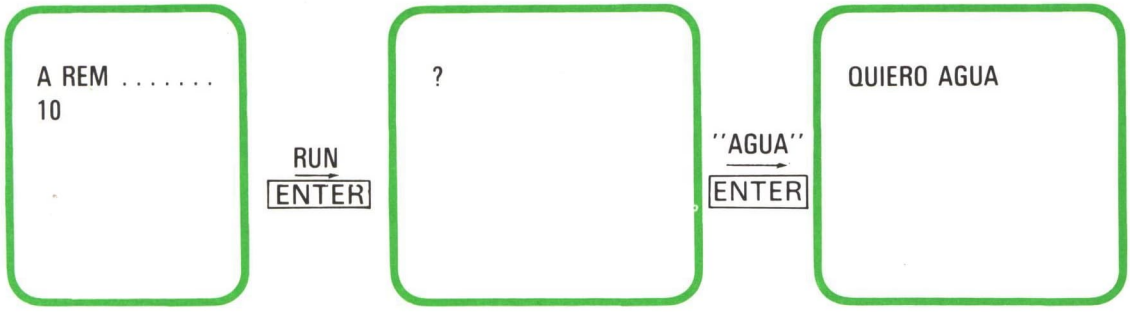
Suma o concatenación de cadenas

La suma o concatenación de cadenas se entiende como la operación que pone una cadena inmediatamente a continuación de la otra sin separación entre ellas. En el programa

```
 1 REM QUE ES LO QUE QUIERE
10 INPUT A$
20 LET B$="QUIERO     "+A$
30 PRINT B$
40 END
```

la instrucción 20 suma la cadena "QUIERO" a otra cadena que se ponga en la variable alfanumérica A$. Se deja un espacio en blanco después de la letra "O" para separar "QUIERO" de la palabra siguiente.

Si al ejecutar este programa introducimos desde el teclado la palabra "AGUA" (con comillas incluidas) al aparecer ?, se obtiene el siguiente resultado:



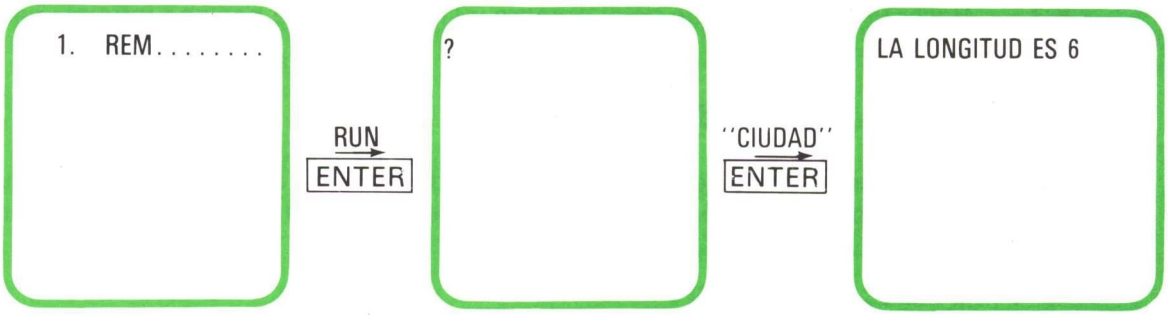
Longitud de una cadena

Se llama longitud de una cadena al *número de caracteres* que tiene. Se calcula con la función LEN (A$) que, en este caso, determina la longitud de la cadena que se guarde en la variable A$.

Por ejemplo, si una vez tecleado el programa

```
 1  REM LONGITUD DE UNA CADENA
10  INPUT A$
20  LET N=LEN (A$)
30  PRINT ''LA LONGITUD ES'' ; N
40  END
```

se ejecuta, y al aparecer ? se introduce la cadena ''CIUDAD'', en la pantalla aparece escrita esta frase: LA LONGITUD ES 6.



Paso de valor numérico a cadena numérica y viceversa

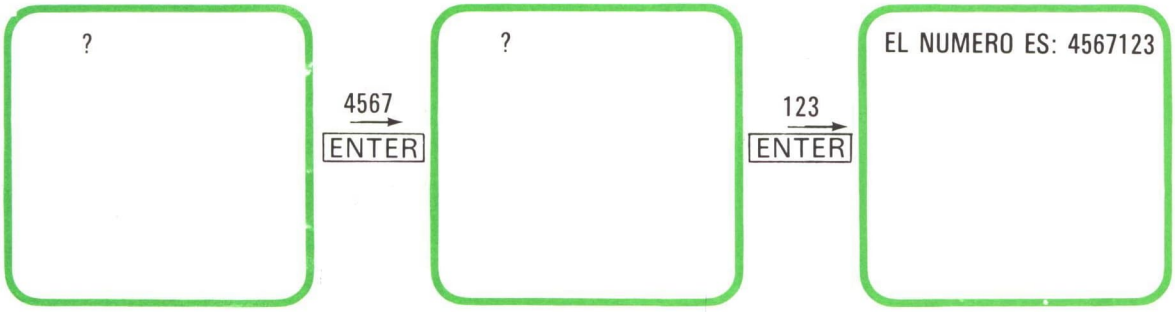
La función STR$ (N) convierte el valor numérico recogido en la variable N en una cadena numérica.

La función VAL(A$) hace la operación inversa, es decir, convierte una cadena numérica, recogida en la variable alfanumérica A$, en un número.

Por ejemplo, al ejecutar el programa

```
 1 REM CONCATENACION DE CADENAS NUMERICAS
10 INPUT N
20 INPUT M
30 LET A$=STR$ (N)
40 LET B$=STR$ (M)
50 LET C$=A$+B$
60 LET P=VAL (C$)
70 PRINT ''EL NUMERO ES: ''; P
80 END
```

si después que aparece en pantalla cada interrogación se introducen los números 4567 y 123, las instrucciones 30 y 40 los convierten en cadena de caracteres, la instrucción 50 los junta o une (no los suma numéricamente puesto que ya no son propiamente números), la instrucción 60 convierte la cadena 4567123 en valor numérico. Finalmente, la instrucción 70 escribe en pantalla: EL NUMERO ES: 4567123.



Extracción de subcadenas

A veces puede interesar extraer de una cadena unos cuantos caracteres consecutivos. Por ejemplo, del nombre CASTELLON extraer sus tres primeras letras: CAS. Esto se puede realizar en BASIC utilizando tres funciones, según que los caracteres que se quieren extraer estén al principio, al final o en medio de la cadena. Así:

LEFT$(A$, N)	Extrae los N *primeros* caracteres de la cadena que se guarda en A$.
RIGHT$(A$, N)	Extrae los N *últimos* caracteres de la cadena A$.
MID$(A$, P, N)	Extrae en la cadena A$, N caracteres a partir del que está en la posición P.

Por ejemplo, si

A$="CASTELLON"	
X$=LEFT$(A$, 3)	guarda en X$="CAS"
Y$=RIGHT$(A$, 4)	guarda en Y$="LLON"
Z$=MID$(A$, 4, 5)	guarda en Z$="TELLO"

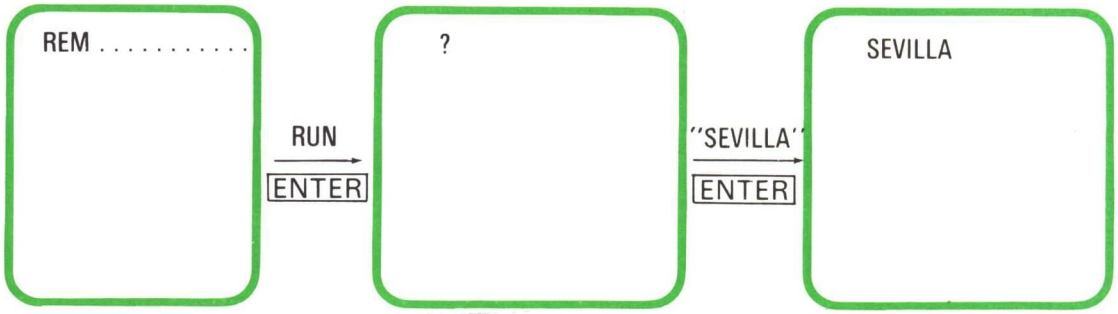
Ejemplo:

Dado el nombre de una provincia, almacenar en X$ sus tres primeras letras, en Y$ las dos últimas y Z$ tres a partir de la segunda letra:

```
 1 REM EXTRACCION DE SUBCADENAS
10 INPUT A$
20 LET X$=LEFT$(A$, 3)
30 LET Y$=RIGHT$(A$, 2)
40 LET Z$=MID$(A$, 2, 3)
50 PRINT X$; Y$; Z$
60 END
```

(en esta instrucción 10 se introduce el nombre de una provincia española)
(se extraen las tres primeras letras de A$)
(se extraen las dos últimas letras de A$)
(se obtiene una subcadena de tres letras a partir de la segunda letra de A$).

Mediante la instrucción 10 se introduce el nombre de una provincia. Con las instrucciones 20, 30, 40 se calculan las funciones, almacenando sus valores en X$, Y$ y Z$, y por último, en la 50 se imprimen estos valores, terminando el programa. Si al ejecutar el programa se introduce desde el teclado la cadena "SEVILLA", en la pantalla aparecerá lo siguiente.



Comparación de cadenas

De la misma forma que una persona puede ordenar alfabéticamente una lista de nombres, el microordenador también puede hacer lo mismo y, además, ordenar cadenas de caracteres.

Internamente el microordenador tiene ordenadas primero las cifras del 0 al 9 y después las letras de la A a la Z.

Al comparar dos cadenas, compara el primer carácter de ambas, después el segundo, etcétera. Así, por ejemplo:

"ARBOL" < "ASTA" (ARBOL *es anterior a* ASTA porque aunque comienzan por la misma letra, la segunda de ARBOL es anterior a la segunda de ASTA.)

"B 30" < "B 312" (B 30 es anterior a B 312 porque aunque los dos primeros caracteres son iguales, sin embargo el tercer carácter 0 es anterior a 1.)

Además de <, se pueden utilizar el resto de los comparadores: >, =, =>, <= y <>, y los operadores lógicos OR, AND y NOT.

Este programa ordena alfabéticamente dos nombres:

```
 1  REM ORDENADOR DE NOMBRES
10  READ A$, B$
20  IF A$ < B$ THEN 50
30  PRINT B$, A$
40  STOP
50  PRINT A$, B$
```

La instrucción 10 lee los dos nombres A$, B$ por medio de READ, luego hay que añadir al programa una línea de datos, por ejemplo:

```
60  DATA PEDRO, JUAN
70  END
```

En la instrucción 20 se comparan estos nombres. Al no verificarse la condición «PEDRO» < «JUAN», se ejecuta la instrucción siguiente, que manda imprimir:

```
JUAN     PEDRO
```

En la 40 se para el proceso.

Si se hubiera cumplido la condición, el control pasaría a la 50 que imprimiría los nombres en el orden contrario.

Para ejecutar el programa hay que teclear, como siempre, RUN [ENTER]

Programa: cambio de moneda

El programa siguiente, que transforma un número de pesetas a dólares, libras o marcos, utiliza la comparación entre cadenas.

```
 1  REM CAMBIO
10  PRINT ''INTRODUZCA EL CAMBIO ACTUAL''
20  PRINT ''UN DOLAR EN PESETAS''
30  INPUT D
40  PRINT ''UNA LIBRA EN PESETAS''
50  INPUT L
60  PRINT ''UN MARCO EN PESETAS''
70  INPUT M
75  CLS
80  STOP
```

Las instrucciones 10 a 70 sirven para introducir el valor en pesetas de cada moneda.

La instrucción 75 CLS borra el contenido de la pantalla. La instrucción 80 STOP detiene la ejecución del programa, justo en ese lugar.

Escribiendo la palabra CONT el programa seguirá en orden secuencial.

```
 90  LET A$=''DOLARES''
100  LET B$=''LIBRAS''
110  LET C$=''MARCOS''
120  PRINT ''CUANTAS PESETAS VA A CAMBIAR''
130  INPUT C
140  PRINT ''A QUE MONEDA QUIERE CAMBIAR''
150  INPUT M$
160  IF M$=A$ THEN 210
170  IF M$=B$ THEN 230
180  IF M$=C$ THEN 250
190  PRINT ''NO SE CAMBIA ESTA MONEDA''
200  GOTO 120
```

Las instrucciones 90 a 100 asignan las cadenas "DOLARES", "LIBRAS" y "MARCOS" a las variables A$, B$ y C$ respectivamente.

De la 120 a la 150 se introduce la cantidad de pesetas y la moneda a la que queremos cambiar estas pesetas.

De la 160 a la 180 se compara M$ con A$, B$ y C$, y como resultado de esta comparación se calcula el número de pesetas (210 a 240).

```
210  LET P=C/D
220  GOTO 260
230  LET P=C/L
240  GOTO 260
250  LET P=C/M
260  PRINT "LA CANTIDAD DE"; M$; "ES"; P
265  WAIT (500)
267  CLS
270  GOTO 120
280  END
```

En la 265 WAIT (500) se da un tiempo de espera (PAUSE (n) en otros microordenadores) y en la 267 se borra la pantalla para posteriormente pedir una nueva cantidad de pesetas.

RESUMEN

Nombre	Significado	Función
A$+B$	Suma o concatenación	La operación + de cadenas forma una nueva cadena con A$ seguida de B$.
LEN (A$)	Longitud	Da el número de caracteres de la cadena A$.
VAL (A$)	Valor numérico	Convierte la cadena numérica A$ en su valor numérico.
STR$ (N)	Obtención de cadena numérica	Convierte el número N en una cadena numérica.
LEFT$ (A$, N)	Subcadena izquierda	Extrae los N primeros caracteres de la cadena A$.
RIGHT$ (A$, N)	Subcadena derecha	Extrae los N últimos caracteres de la cadena A$.
MID$ (A$, P, N)	Subcadena interior	Extrae los N caracteres de la cadena A$ desde la posición P.
A$ < B$	Comparación	La cadena A$ es anterior a la B$.
CLS	Borra	Borra la pantalla.
WAIT(n), PAUSE(n)...	Espera, pausa...	Detiene la ejecución del programa un tiempo, que depende de **n**.

EJERCICIOS DE RECAPITULACION

6.1. ¿Qué caracteres imprime el siguiente programa?

```
10 LET A$="FERNANDEZ"
20 PRINT LEFT$ (A$, 2)
30 PRINT RIGHT$ (A$, 1)
40 PRINT MID$ (A$, 2, 3)
```

6.2. Escribir las instrucciones necesarias para imprimir el número de caracteres de la palabra NABUCODONOSOR.

6.3. ¿Qué imprimirá el siguiente programa?

```
10 LET A$="PATA"
20 LET B$="TA"
30 PRINT A$+B$
40 END
```

6.4. ¿Qué instrucciones son correctas? ¿Por qué?

a) IF X="FECHA" THEN 500
b) IF N$ < > A+B THEN 600
c) IF P$="1234" THEN 300
d) IF G$="MAYOR" THEN 220

6.5. ¿Qué tiene que ocurrir para que se cumpla la condición P$ < Q$?

6.6. Escribir una o varias instrucciones BASIC para representar la condición: Si N$="SI" transferir el control a la posición 70, y si N$="NO" transferirlo a la 90.

6.7. Sea una cadena A$ en la que están incluidos el primer apellido de un alumno, a continuación el segundo apellido y después el nombre, cada uno con 20 caracteres (si no se ocupan se dejan con blancos); además está incluida la edad (2 caracteres). Escribir un programa que imprima por separado cada una de estas características.

6.8. Escribir un programa tal que para tres personas cuyos apellidos, nombre y edad se conocen y están incluidos en tres cadenas A$, B$ y C$, imprima en pantalla su edad media (ver el ejercicio anterior y usar la instrucción VAL (A$)).

6.9. Escribir un programa que introduzca por separado el día, el mes y el año de nacimiento de una persona, forme una única cadena para estos datos e imprima el resultado.

6.10. Escribir un programa que permita introducir una palabra de cinco letras y después la imprima de atrás hacia delante.

6.11. Escribir un programa que lea una serie de palabras de cuatro letras e imprima solamente las que empiecen por la letra A.

6.12. Escribir un programa que lea una serie de palabras de cuatro letras e imprima solamente las que empiecen con la letra A y terminen con la S.

6.13. Dadas las variables:

```
A$="EL"
B$="DIA"
C$="COMIENZA"
D$="AL"
E$="AMANECER"
```

obtener la concatenación de estas variables e imprimir el resultado.

6.14. Rectificar el programa anterior para que al imprimir deje un espacio en blanco.

6.15. Realizar un programa que lea tres cadenas y dé la longitud total de las tres.

6.16. Construir un programa que lea una cadena, calcule su longitud y vaya extrayendo subcadenas desde la izquierda, de longitud 1, longitud 2..., longitud máxima.

6.17. Leer una cadena y añadir blancos (espacios en blanco) hasta que alcance longitud 20.

6.18. Diseñar un programa que no tenga fin y al leer una cadena imprima el primer y el último carácter.

6.19. Escribir un programa que compruebe si una cadena contiene la letra A.

6.20. Realizar un programa que dé el informe «CORRECTO» si la cadena leída tiene longitud inferior a 3 y que imprima «MAL» en caso contrario.

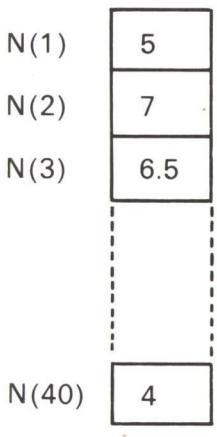
7 Listas y tablas

Listas (vectores)

Las variables numéricas y alfanuméricas se pueden introducir en la memoria del ordenador en forma de tablas.

Así, si se quiere almacenar en la memoria las notas por asignatura de cuarenta alumnos de una clase, con vista a una posterior impresión, escribiríamos:

```
10  DIM N(40)
20  FOR I=1 TO 40
30  INPUT N(I)
40  NEXT I
50  END
```



La instrucción 10 DIM N(40) (DIM deriva de DIMENSION) sirve para especificar las dimensiones de la tabla que se va a introducir, reservando 40 posiciones en la memoria. Algunos microordenadores empiezan a contar las posiciones de memoria a reservar desde cero, luego en total reservarían 41 posiciones. Al ejecutar este programa se interrumpe en la instrucción 30, apareciendo en la pantalla una ?. Introduciendo desde el teclado un número, continúa la ejecución, pero seguirá interrumpiéndose hasta que se hayan introducido las 40 notas.

La instrucción DIM ( ) no es imprescindible en algunos microordenadores si se van a usar listas con un número de elementos menores que 10. Sin embargo, si no se utilizara, cuando el número de elementos fuera mayor que 10, el microordenador imprimiría en pantalla un mensaje de error.

Se puede comprobar que los datos están en la memoria dando un PRINT, y así, si el tercer INPUT N(3) fuera un 6.5, al escribir PRINT N(3) y ENTER aparece en la pantalla el número 6.5.

Si se quiere imprimir todas las notas, se escribe el programa siguiente llevando la instrucción 50 END al final con el número 130 y después se ejecuta:

```
100  FOR I=1 TO 40
110  PRINT N(I)
120  NEXT I
130  END
```

Aparecen en la pantalla los números introducidos uno debajo de otro:

```
 5
 7
6.5
 .
 .
 4
```

El programa anterior se puede sustituir por el siguiente, obteniéndose el mismo resultado:

```
 1  DIM N(40)
10  FOR I=1 TO 40
20  READ N(I)
30  NEXT I
40  DATA 5, 7, 6.5, ..., 4
50  END
```

La ventaja de este programa respecto al anterior es que permite más autonomía al usuario, pues ahora puede introducir mediante la instrucción 40 todas las notas de una sola vez, antes de la ejecución. Esta, además, es mucho más rápida, en cuanto que no se interrumpe ni una sola vez.

Para comprobar que este programa produce el mismo efecto que el anterior, se puede ampliar con las instrucciones

```
50  FOR I=1 TO 40
60  PRINT N (I)
70  NEXT I
80  END
```

llevando la instrucción END al final del programa. Al ser ejecutado imprime los valores de la tabla N(I) almacenada en la memoria.

Ordenando números

Se tienen N números y se desea ordenarlos de menor a mayor.

Una vez introducidos los números en la memoria, tendremos que comparar los números situados en lugares contiguos; por ejemplo, el S(J) con el S(J+1).

Si S(J) es mayor que S(J+1), invertiremos los puestos. Esto lo hacemos para cada par de elementos y, repitiendo el proceso N−1 veces, llegaremos a tenerlos totalmente ordenados. El proceso esbozado se puede realizar por medio de bucles aislados, según un programa como el que sigue:

```
 1  REM ORDENACION DE NUMEROS
 5  DIM S(100)
10  INPUT N
20  FOR I=1 TO N
30  INPUT S(I)
40  NEXT I
```

Con la instrucción 10 se introduce el número N, que da el tamaño de la tabla; éste no podrá ser mayor que 100. Mediante la instrucción 30 se introducen desde el teclado los números S(I) de la tabla que se desea ordenar.

Con las instrucciones siguientes, el programa ordena los números de la tabla.

```
100  FOR I=1 TO N-1
110  FOR J=1 TO N-1
120  LET X=S(J)
130  LET Y=S(J+1)
140  IF X<=Y THEN 170
150  LET S(J)=Y
160  LET S(J+1)=X
170  NEXT J
180  NEXT I
```

Las instrucciones 120 y 130 comparan los números S(J) y S(J+1), y si están en orden (140 IF X<=Y) los guarda así. Si no están ordenados hace el cambio de orden en 150 y 160 y los guarda así en la memoria. Este proceso se repite para cada valor de I, desde I=1 hasta N−1. Al ejecutarse el programa, las instrucciones 300 a 320 hacen aparecer en pantalla los números de la tabla uno debajo de otro en orden creciente.

```
300  FOR I=1 TO N
310  PRINT  S (I)
320  NEXT I
330  END
```

Verificación del programa

Sean los cuatro números siguientes: 8, 5, 7, 3.

La instrucción 10 pide el número N de elementos a comparar; en este caso, 4. El bucle 20-40 introduce los cuatro valores en memoria; por ejemplo, 8, 5, 7 y 3.

Con los bucles anidados que se encuentran en las instrucciones 100 a 180 se consigue ordenar de menor a mayor los cuatro números.

En el bucle interior, la variable J toma sucesivamente los valores 1, 2 y 3, y en el exterior, la variable I también toma los mismos valores.

S(1)	8
S(2)	5
S(3)	7
S(4)	3

A continuación se analiza paso a paso el programa:

Para I=1

Si J=1

X=S(1)=8

Y=S(2)=5

Como no se cumple la condición expresada en 140, hace S(1)=Y [=5] y S(2)=X [=8]

Si J=2

X=S(2)=8

Y=S(3)=7

que tampoco cumple la condición expresada en 140, luego

S(2)=Y [=7]

S(3)=X [=8]

Si J=3

X=S(3)=8

Y=S(4)=3

Tampoco cumple, luego

S(3)=Y [=3]

S(4)=X [=8]

Luego para I=1 la tabla quedaría

S(1)	5
S(2)	7
S(3)	3
S(4)	8

Para I=2

J=1 S(1)=5 S(2)=3

J=2 S(2)=3 S(3)=7

J=3 S(3)=7 S(4)=8

S(1)	5
S(2)	3
S(3)	7
S(4)	8

Para I=3

J=1	S(1)=3	S(2)=5
J=2	S(2)=5	S(3)=7
J=3	S(3)=7	S(4)=8

S(1)	3
S(2)	5
S(3)	7
S(4)	8

Al ejecutar el programa (RUN [ENTER]) aparecen escritos en la pantalla los cuatro números de la tabla, uno debajo de otro, en orden creciente

```
3
5
7
8
```

Ejercicio

7.1. Probar el programa para una mayor cantidad de números, por ejemplo 10, y con algunos números repetidos.

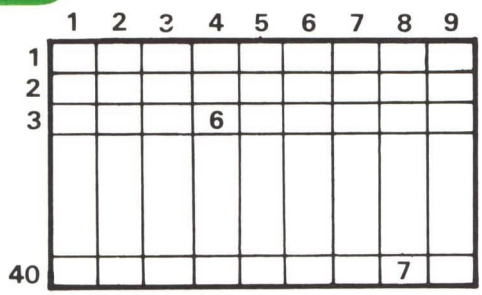
Tabla de dos dimensiones (matrices)

También pueden almacenarse datos en la memoria en forma de tablas de dos dimensiones (matrices).

Así, para almacenar las notas de 40 alumnos en 9 asignaturas en forma de tabla de 40 filas y 9 columnas, se escribiría este programa:

```
10  DIM N (40, 9)
20  FOR F=1 TO 40
30  FOR C=1 TO 9
40  INPUT N (F, C)
50  NEXT C
60  NEXT F
70  END
```

Aquí N es el nombre de la tabla, F el número de filas y C el número de columnas. En la memoria tendríamos:



Por ejemplo, la nota de valor 6 está en la posición (3, 4) y corresponde a N (3, 4), es decir, es la nota obtenida por el alumno número 3 en la asignatura identificada con el número 4.

En cambio, la nota de valor 7 está en la posición (40, 8) y corresponde a N (40, 8), es decir, al alumno número 40 y en la asignatura identificada con el número 8.

Para evitar introducir las $40 \times 9 = 360$ notas en el proceso de ejecución mediante la instrucción 40 INPUT N (F, C), lo cual supone una ejecución muy lenta, se puede transformar el programa anterior utilizando las instrucciones READ y DATA.

```
 10  DIM N (40, 9)
 20  FOR F=1 TO 40
 30  FOR C=1 TO 9
 40  READ N(F, C)
 50  NEXT C
 60  NEXT F
 70  DATA --, --, --, --, ...
 80  DATA --, --, --, --, ...
100  END
```

El resultado sería similar siempre y cuando se dispongan adecuadamente los datos de las sentencias DATA (70 y 80). Para comprobarlo, se obliga al programa a imprimir la tabla, añadiendo las instrucciones 200 a 240 y trasladando la instrucción 100 END al final, precedida del número 250.

```
200  FOR F=1 TO 40
210  FOR C=1 TO 9
220  PRINT N(F, C)
230  NEXT C
240  NEXT F
```

Ejercicios

7.2. Los precios de cinco productos en tres ciudades españolas están dados en la siguiente tabla.

	Valencia	*Las Palmas*	*Soria*
Naranjas	100	140	120
Plátanos	100	90	130
Fresones	90	150	110
Manzanas	80	100	90
Peras	60	90	90

Hacer un programa que dé los precios medios de los distintos productos.

Programa: suma de las filas y columnas de una tabla

Este programa suma las filas y columnas de una tabla que se ha introducido previamente en el microordenador:

```
 1  REM SUMA FILAS Y COLUMNAS
10  READ M, N
15  DIM T(M, N)
17  DIM S(M)
19  DIM H(N)
25  FOR I=1 TO M
30  FOR J=1 TO N
40  READ T(I, J)
50  NEXT J
60  NEXT I
```

Para introducir la tabla se leen los datos contenidos en las instrucciones DATA (500 y 510). Previamente se definen las variables que se van a necesitar en el programa.

```
100  PRINT ''SUMA DE COLUMNAS DE CADA FILA''
110  PRINT
120  LET S1=0
130  FOR I=1 TO M
140  LET S(I)=0
150  FOR J=1 TO N
160  LET S(I)=S(I)+T(I, J)
170  NEXT J
180  PRINT ''FILA''; I, ''SUMA= ''; S(I)
```

Para cada fila (I) se suman las columnas J=1 a N acumulando los resultados en la variable S(I), imprimiendo después el resultado.

```
190  LET S1=S1+S(I)
200  NEXT I
210  PRINT
220  PRINT
230  PRINT ''SUMA DE LA SUMA DE LAS FILAS= ''; S1
240  PRINT
```

A continuación, en la instrucción 190, se van sumando las sumas correspondientes a cada fila, imprimiéndose el resultado.

```
300  PRINT "SUMA DE FILAS DE CADA COLUMNA"
310  PRINT
320  LET H1=0
330  FOR J=1 TO N
340  LET H(J)=0
350  FOR I=1 TO M
360  LET H(J)=H(J)+T(I, J)
370  NEXT I
380  PRINT "COLUMNA"; J, "SUMA= "; H(J)
390  LET H1=H1+H(J)
400  NEXT J
410  PRINT
420  PRINT "SUMA DE LAS SUMAS DE COLUMNAS"; S1
```

Manteniendo fija cada columna se suman todas las filas (I=1 a M) acumulando su suma en la variable H(J), suma que se imprime en la instrucción 380. Después se van sumando las sumas correspondientes a cada columna, imprimiéndose el resultado a continuación.

```
500  DATA --, --, -
510  DATA --. --. -
520  END
```

Tablas de caracteres alfanuméricos

También se pueden colocar en la memoria caracteres alfanuméricos en forma de tabla. Así, la introducción de 40 apellidos de los alumnos de una clase, y su almacenamiento en memoria, se conseguiría con este programa:

```
10  DIM A$(40, 20)
20  FOR I=1 TO 40
30  INPUT A$(I)
40  NEXT I
50  END
```

La instrucción 10 DIM A$ (40, 20) reserva en la memoria 40 lugares para introducir como máximo 20 caracteres alfanuméricos en cada posición (en muchos BASIC no hace falta poner este límite).

En la memoria quedaría almacenada una tabla como la siguiente:

A(1)	FERNANDEZ
A(2)	LOPEZ
A(3)	PEREZ
	.....
A(40)	SANCHEZ

También aquí se podrían haber usado las instrucciones READ y DATA como alternativa al INPUT.

Para que el programa llegara a escribir los apellidos en la pantalla habría que añadir estas instrucciones y después ejecutarlo. Es obvio que la instrucción 50 END habría que trasladarla al final, precedida del número 140.

```
100  FOR I=1 TO 40
110  PRINT A$(I)
120  NEXT I
130  END
```

Ejercicio

7.3. Construir programas que introduzcan en la memoria y después imprima:

a) los meses del año,

b) los días de la semana.

Ordenación de cadenas

El programa siguiente ordena alfabéticamente una lista de cuatro nombres:

```
  1  REM ORDENADOR DE NOMBRES
 10  DIM A$(4)
 20  FOR I=1 TO 4
 30  READ A$(I)
 40  NEXT I
 50  FOR J=1 TO 3
 60  FOR K=J+1 TO 4
 70  IF A$ (J) < A$(K) THEN 110
 80  LET X$=A$(J)
 90  LET A$(J)=A$(K)
100  LET A$(K)=X$
110  NEXT K
120  NEXT J
130  FOR I=1 TO 4
140  PRINT A$(I)
150  NEXT I
```

El bucle 20-40 lee los cuatro nombres, y como aparece la instrucción READ hay que añadir al programa una línea de datos, por ejemplo:

```
160  DATA PEDRO, JUAN, ANTONIO, CARLOS
170  END
```

El bucle 50-120 consigue que para J=1 se almacene en A(1) el nombre que es primero alfabéticamente, gracias al bucle interno 60-110 que va comparando A(1) con sus siguientes; de tal forma que si alguno es anterior que él se intercambian en las instrucciones 80, 90 y 100. Para J=2 almacena en A$(2) el primero en orden alfabético de A$(2), A$(3) y A$(4). Y por último, para J=3, en A$(3) se almacena el primero en orden alfabético de A$(3) y A$(4).

Seguimiento del programa

1.º Al principio: A$(1)="PEDRO"
A$(2)="JUAN"
A$(3)="ANTONIO"
A$(4)="CARLOS"

2.º J=1.
K=2.

Al no cumplirse A$(1) < A$(2), se ejecuta la instrucción 80:

```
 80 X$="PEDRO"
 90 A$(1)="JUAN"
100 A$(2)="PEDRO"
```

con lo que se han intercambiado A$(1) y A$(2)

```
A$(1)="JUAN"
A$(2)="PEDRO"
A$(3)="ANTONIO"
A$(4)="CARLOS"
```

3.º J=1
K=3.

Tampoco se cumple A$(1) < A$(3), luego se ejecuta 80:

```
 80 X$="JUAN"
 90 A$(1)="ANTONIO"
100 A$(3)="JUAN"
```

```
A$(1)="ANTONIO"
A$(2)="PEDRO"
A$(3)="JUAN"
A$(4)="CARLOS"
```

4.º J=1
K=4.

Como A$(1) < A$(4) se termina el bucle para J=1.

5.º J=2
K=3.

No se verifica que A$(2) < A$(3) y entonces se ejecuta 80:

```
 80 X$="PEDRO"
 90 A$(2)="JUAN"
100 A$(3)="PEDRO"
```

```
A$(1)="ANTONIO"
A$(2)="JUAN"
A$(3)="PEDRO"
A$(4)="CARLOS"
```

6.º J=2
K=4.

No es cierto que A$(2) < A$(4), por lo que se ejecuta 80:

```
 80 X$="JUAN"
 90 A$(2)="CARLOS"
100 A$(4)="JUAN"
```

```
A$(1)="ANTONIO"
A$(2)="CARLOS"
A$(3)="PEDRO"
A$(4)="JUAN"
```

Se termina el bucle para J=2.

7.º J=3
K=4.

No se cumple A$(3) < A$(4). Se ejecuta 80:

```
 80 X$="PEDRO"
 90 A$(3)="JUAN"
100 A$(4)="PEDRO"
```

```
A$(1)="ANTONIO"
A$(2)="CARLOS"
A$(3)="JUAN"
A$(4)="PEDRO"
```

8.º Al terminarse el bucle K, se ejecuta el bucle 130–150 que imprime los nombres ordenados alfabéticamente:

```
ANTONIO
CARLOS
JUAN
PEDRO
```

EJERCICIOS DE RECAPITULACION

7.1. En la memoria de un microordenador se tienen estas variables con sus correspondientes valores iniciales:

Q	2	A(1)	37
A	3	A(2)	4
A1	1	A(3)	23
		A(4)	19

Escribir el valor de las siguientes variables: A(A), A(A1), A(A(Q)), A(A(2)).

7.2. Señalar las variables de dos dimensiones, correctas:

a) X(2+2).
b) X(5, 5).
c) X(A+B, C)
d) X(X(1, 2), X(2, 1)).
e) X(A, A).

7.3. A la siguiente tabla se asigna el nombre A

	1	2
1	1	2
2	3	4
3	5	6

a) ¿Qué dimensiones tiene A?
b) ¿Qué valores tienen A(1, 1) y A(3, 1).
c) Si X=2, e Y=3, ¿cuál es el valor de A(X, Y), A(X+1, Y−1) y A(A(1. 2), A(2, 1)−1)?

7.4. ¿Qué aparecerá en pantalla al ejecutar el siguiente programa?

```
10 READ N
20 FOR K=1 TO N
30 READ X(K)
40 NEXT K
50 FOR K=1 TO N
60 IF X (K) < 0 THEN 80
70 PRINT X(K)
80 NEXT K
90 DATA 7
100 DATA 23, -44, 37, 0, -15, -38, 14
200 END
```

7.5. Si se añadiesen al programa anterior las siguientes instrucciones

```
110 DATA 8,4, -3,2, 2,4, 3,8, 5,1
120 DATA 5,0, -2,4, 3,9, -8., 4.3, 8.4
```

¿qué problemas se presentarían? ¿Cómo se solucionarían?

7.6. Escribir un programa que introduzca en cada elemento de una tabla de 10 filas y 5 columnas: *a)* 0; *b)* 1.

7.7. Escribir varias instrucciones que permitan escribir en una variable A$(I) los meses del año.

7.8. Escribir varias instrucciones que permitan escribir en una variable A$(I) los días de la semana.

7.9. Escribir varias instrucciones que permitan calcular la suma de los 100 primeros elementos de una lista T(200).

7.10. Al hacer una encuesta para anticipar los posibles resultados de unas próximas elecciones se han hecho las siguientes preguntas:

a) ¿A qué candidato votaría?

1. NOMBRE 1
2. NOMBRE 2
3. EN BLANCO.

b) ¿En qué grupo de edad está usted?

1. Menos de 30 años.
2. 30 años o más.

Escribir un programa que haga el recuento de los votos (que se introduzcan con instrucciones **READ** y **DATA**) y los clasifique por edades. [Se puede definir una variable A (V, E) siendo V el voto y E el intervalo de edad del votante. Así, A (1, 1) es un voto conseguido por el candidato 1 de un votante menor de 30 años.]

7.11. Dado un conjunto de valores, elaborar un programa que escriba aquellos que se encuentran entre el menor y el mayor.

7.12. Se tiene una lista de 20 personas entre los 30 y 69 años. Hacer un programa que contabilice el número de personas que se encuentran entre

30 — 39
40 — 49
50 — 59
60 — 69

Imprimir los resultados.

7.13. Dadas las calificaciones de 10 alumnos en cinco asignaturas, escribir un programa que obtenga la nota media de cada alumno.

7.14. Ampliar el programa correspondiente al ejercicio anterior, para que dé también la media por asignatura.

7.15. Sea una lista de 20 elementos, los cuales pueden ser valores positivos, negativos o nulos. Hacer un programa que imprima cuántos elementos son positivos, cuántos negativos y cuántos nulos.

7.16. Una red comercial tiene 10 tiendas y cada tienda 4 departamentos. Escribir un programa que lea los valores de las ventas por tienda y departamento e imprima:

1.º El total de ventas por tienda.
2.º El total de ventas por departamento en todas las tiendas.

8 Funciones y subrutinas

Algunas funciones del lenguaje BASIC

Además de los operadores *, /, +, − y ↑, y la raíz cuadrada SQR(X), que ya hemos visto, el lenguaje BASIC permite utilizar otro conjunto de funciones, algunas de las cuales aparecen en el siguiente programa.

```
10  INPUT X
20  PRINT "X", X
30  PRINT "ABS(X)", ABS(X)
40  PRINT "INT(X)", INT(X)
50  PRINT "SGN(X)", SGN(X)
60  END
```

Al ejecutar RUN [ENTER], si se introduce el valor X=−7.8 se obtiene en la pantalla lo siguiente:

```
X          −7.8
ABS(X)      7.8
INT(X)     −8
SGN(X)     −1
```

Si se introduce el valor X=1.3 se obtiene:

```
X           1.3
ABS(X)      1.3
INT(X)      1
SGN(X)      1
```

La función ABS(X) da el valor absoluto de X.
La función INT(X) da como valor el entero más pequeño próximo a X.

Así, INT (7.8)=7 y INT (−7.8)=−8.

La función SGN(X) da +1 si X es positivo, −1 si X es negativo y 0 si X es 0.

Aplicación: cambio de unidades

El programa siguiente muestra una utilización muy frecuente de la función INT:

```
 1 REM PASA DE METROS A KILOMETROS
10 INPUT M
20 PRINT M; ''METROS''
30 LET K=INT (M/1.000)
40 LET M=M-K * 1.000
50 PRINT K; ''KM''
60 PRINT M; ''M''
70 END
```

Así, para M=1.850 metros aparecerá en pantalla:

```
1.850 METROS
    1 KM
  850 M
```

— La instrucción 30 almacena en K el cociente entero de la división entre M y 1.000.
— La instrucción 40 almacena en M el resto de la división entre M y 1.000. (Recordar: D=dc+r, de donde r=D−dc.)

Aplicación: redondeo de números

Para redondear un número a un número de cifras decimales determinado, por ejemplo, dos, puede usarse el siguiente programa:

```
 10 REM REDONDEO
 20 PRINT ''INTRODUZCA UN NUMERO''
 30 INPUT A
 40 LET A1=INT (A * 100+0.5)/100
 50 PRINT ''EL NUMERO REDONDEADO ES''; A1
 60 PRINT
 70 GOTO 20
100 END
```

Para A=0.3333 la instrucción 40 LET A1=INT (A * 100+0.5)/100 efectuará:

A * 100=33.33
A * 100+0.5=33.83
INT (A * 100+0.5)=33
INT (A * 100+0.5)/100=0.33

Así aparecerá en pantalla después de escribir RUN [ENTER]. De la misma manera, para A=0.6666 la instrucción 40 efectuará:

A * 100=66.66
A * 100+0.5=67.16
INT (A * 100+0.5)=67
INT (A * 100+0.5)/100=0.67

Así, después de escribir RUN [ENTER] aparecerá en pantalla 0.67.

Ejercicios

8.1. Hacer un programa que pase segundos a horas.

8.2. Hacer un programa que pase grados a radianes.

8.3. Hacer un programa que pase calorías a julios.

8.4. Escribir un programa que permita redondear:

a) a una cifra decimal,
b) a tres cifras decimales,
c) a **n** cifras decimales.

Funciones trigonométricas, exponenciales y logarítmicas

Otras funciones importantes que posee el BASIC son las funciones trigonométricas: seno de x, SIN (X), coseno de x, COS (X), y tangente de x, TAN (X), en las que X debe estar siempre expresado en radianes. También posee la función recíproca arcotangente de x, ATN (X), que da el ángulo correspondiente a una tangente de valor x en radianes.

El siguiente programa imprime el seno y coseno de ciertos ángulos expresados en grados, transformándolos previamente en radianes

```
 1  REM SENO Y COSENO
10  PRINT "INTRODUZCA UN ANGULO EN GRADOS"
20  INPUT X
30  PRINT X; "GRADOS"
40  LET F=3.14159265/180
50  PRINT "SEN (X)", SIN (X * F)
60  PRINT "COS(X)", COS(X * F)
70  END
```

En la instrucción 30 se ha introducido el factor F para pasar de grados a radianes que después se utiliza en las instrucciones 50 y 60. (Recordar que $\alpha_{rad}=\frac{\alpha^{o}\pi}{180^{o}}$, donde $F=\frac{\pi}{180^{o}}$; es decir, $\alpha_{rad}=\alpha F$.)

Al ejecutar el programa, si se introduce X=30 se obtiene en pantalla

```
30 GRADOS
SEN(X) 0.5
COS(X) 0.866025
```

y para X=60° se obtiene

```
60 GRADOS
SEN(X) 0.866025
COS(X) 0.5
```

Además, el lenguaje BASIC posee las funciones exponencial, EXP(X), y logarítmica, LOG(X). La función exponencial eleva el número *e* a un exponente X. Ejemplo:

EXP (0.5) equivale a $e^{0,5}=e^{1/2}=\sqrt{e}$

La función logarítmica calcula el logaritmo natural o neperiano de x, siendo $x>0$. Ejemplo:

LOG (52.23) es equivalente a ln 52.23.

Ejercicios

8.5. Las fórmulas del ángulo doble son:

$\operatorname{sen} 2x=2\operatorname{sen} x \cos x$
$\cos 2x=\cos^2 x-\operatorname{sen}^2 x$

Escribir un programa que imprima estos valores.

8.6. Diseñar un programa que dé un mensaje si el seno de un ángulo determinado es mayor que el coseno.

8.7. Hacer un grupo que imprima el logaritmo y el exponencial de los diez primeros números naturales.

RESUMEN DE FUNCIONES

Función	Valor
SGN(X)	+1 si X > 0 0 si X = 0 −1 si X < 0
ABS(X)	X si X ≥ 0 −X si X < 0
SQR(X)	$\sqrt{X}$ si X ≥ 0 error si X < 0
INT(X)	parte entera de X
LOG(X)	log(x) si X > 0 error si X = 0 error si X < 0
EXP(X)	e^x (donde e = 2.71828)
SIN (A)	seno de A (A en radianes)
COS (A)	coseno de A (A en radianes)
TAN (A)	tangente de A (A en radianes)
ATN (T)	arco cuya tangente es T

La función azar

Existen fenómenos que son aleatorios. Un fenómeno, proceso o experimento es aleatorio si de antemano no se puede conocer con certeza el resultado. Por ejemplo, el lanzamiento de un dado es un experimento aleatorio porque no se puede predecir qué resultado va a salir. Otro dispositivo que permite obtener un número aleatorio cada vez que se detiene, es la ruleta.

También un microordenador es capaz de generar números al azar con la ayuda de la función RND(A), que deriva de RANDOM (azar). La letra entre paréntesis representa un argumento, que es un número determinado.

El programa

```
10  FOR I=1 TO 5
20  LET X=RND(1)
30  PRINT X
40  NEXT I
50  END
```

que imprime en pantalla los cinco números

```
0.123543
0.233798
0.897254
0.547232
0.798401
```

ha generado para cada I (=1, 2, 3, 4, 5) en la instrucción 20 un número al azar entre 0 y 1, excluido este último (*).

Aplicación: simulación de experimentos aleatorios

Con la instrucción RND(A) se puede simular un fenómeno aleatorio, por ejemplo, el lanzamiento de una moneda. En este lanzamiento caben dos resultados posibles: CARA, al que le asignamos el número 1, y CRUZ, al que le asignamos el número 0. Para simular estos dos resultados transformamos la función RND(1).

RND(1) da un número aleatorio entre 0 y 1
RND(1)+0.5 da un número aleatorio entre 0.5 y 1.5
INT(RND(1)+0.5) puede tomar aleatoriamente cada vez dos valores: 0 y 1;

Por ejemplo, si el número aleatorio producido por X=RND(1) fuera 0.342701, se tendría: X+0.5=0.842701 y INT(0.842701)=0. Si el número aleatorio hubiera sido 0.732211, la instrucción INT(RND(1)+0.5) daría la parte entera de (0.732211+0.5)=parte entera (1.232211)=1.

(*) Algunos microordenadores, para conseguir un azar de las características del que se consigue con RND(A), necesitan dos instrucciones: RAND (randomize) y RND, sin argumento. Por ejemplo, el programa anterior necesitaría escribirse así:

```
 5  RAND
10  FOR I=1 TO 5
20  LET X=RND
30  PRINT X
40  NEXT I
50  END
```

Este programa simula el lanzamiento de una moneda.

```
 1  REM SIMULACION DEL LANZAMIENTO DE UNA MONEDA
 5  LET D=0
10  LET C=0
15  INPUT N
20  FOR I=1 TO N
25  LET X=INT (RND(1)+0.5)
30  IF X=1 THEN 70
40  PRINT ''CRUZ''
50  LET D=D+1
60  GOTO 90
70  PRINT ''CARA''
80  LET C=C+1
90  NEXT I
100 PRINT ''HAN SALIDO''; C; ''CARAS''
110 PRINT ''HAN SALIDO''; D; ''CRUCES''
120 END
```

Si N fuera lo suficientemente grande, es decir, las instrucciones entre la 20 y la 90 se ejecutaran muchas veces, el número de caras que se obtendría sería muy próximo al número de cruces, que es lo que ocurre cuando se hace el experimento real un número muy elevado de veces. Es decir, este programa simula el fenómeno aleatorio del lanzamiento de una moneda N veces.

De manera semejante podemos simular *el lanzamiento de un dado.*

Para que los números entre 1 y 6 se obtengan con la misma probabilidad debemos escribir:

```
10  INPUT N
20  FOR I=1 TO N
30  LET A=INT (RND(1) * 6+1)
40  PRINT A
50  NEXT I
60  END
```

La instrucción 30 LET A=INT(RND(1) * 6+1) genera un número al azar entre 1 y 6, ya que

— RND(1) genera un número entre 0 y 1, excluido éste;

— RND(1) * 6, genera un número entre 0 y 6, excluido éste;

— RND(1) * 6+1 genera un número entre 1 y 7, excluido éste;

— INT (RND(1) * 6+1), al tomar la parte entera, genera un número entre 1 y 6, éste incluido.

Ejercicio

8.8. Generar números al azar:

— Entre 0 y 9.
— Entre 0 y 100.
— Entre 10 y 20.

Definición de funciones

Además de las funciones anteriores, se pueden introducir en BASIC veintiséis funciones, desde FNA(X) hasta FNZ(X), por medio de una definición (DEF). Así, el programa:

```
10  DEF FNA(X)=2 * X ↑ 2+5 * X+8
20  FOR X=1 TO 9 STEP 2
30  PRINT X, FNA (X)
40  NEXT X
50  END
```

hace aparecer en pantalla

```
1      15
3      41
5      83
7     141
9     215
```

Es decir, siempre que FNA(X) aparece en el programa toma los valores que resultan de sustituir los diversos valores de X en la expresión correspondiente a FNA(X).

Cada DEF introduce una sola función desde FNA( ) hasta FNZ( ). La expresión a la derecha del signo igual puede contener cualquier fórmula, incluyendo las funciones SIN (X), COS (X), etc.

DEF FNY(X)

Formato: DEF FNY(X)=expresión matemática.
Función: define la función FNY(X)=expresión matemática.

Además hay algunos microordenadores que pueden definir funciones de varias variables, por ejemplo:

```
 1  REM SUMA DE TRES DADOS
10  DEF FNA(X, Y, Z)=X+Y+Z
20  FOR X=1 TO 6
30  FOR Y=1 TO 6
40  FOR Z=1 TO 6
50  PRINT X, Y, Z, FNA (X, Y, Z)
60  NEXT Z
70  NEXT Y
80  NEXT X
90  END
```

En la instrucción 10 se define la función FNA(X, Y, Z)=X+Y+Z. Por medio de los tres bucles 20-80, 30-70 y 40-60 se van generando los distintos resultados posibles. Por último, los valores de los puntos correspondientes a los tres dados y su suma son impresos por medio de la instrucción 50.

Al ejecutar este programa, tecleando RUN ENTER, se obtiene el siguiente resultado (en una pantalla de 4 zonas):

```
 1          1          1          3
 1          1          2          4
 1          1          3          5
 1          1          4          6
 1          1          5          7
 1          1          6          8
 1          2          1          4
 1          2          2          5
 1          2          3          6
 1          2          4          7
 1          2          5          8
 1          2          6          9
 1          3          1          5
 .          .          .          .
 .          .          .          .
 .          .          .          .
 6          6          4          16
 6          6          5          17
 6          6          6          18
```

De la misma forma que se han definido funciones numéricas, se pueden definir también funciones de caracteres, desde FNA$(X) hasta FNZ$(X).

El siguiente programa extrae el primer carácter de cada cadena y los concatena (suma de cadenas):

```
 1  REM SUMA DE INICIALES
10  DEF FNA$(X$,Y$)=LEFT$(X$,1)+LEFT$ (Y$, 1)
20  READ X$, Y$
30  PRINT X$, Y$, FNA$(X$, Y$)
40  DATA ARBOL, CALLE
50  END
```

En la instrucción 10 se define la función FNA$(X$, Y$), que extrae el primer carácter de X$ e Y$ y los concatena (suma de caracteres).

La instrucción 20 lee dos cadenas X$ e Y$ y en la 30 se imprimen éstas y el resultado de aplicarlas la función FNA$(X$, Y$).

Por último, la 40 da los datos ARBOL Y CALLE.

Si ejecutamos el programa (RUN [ENTER]) se obtiene, en una pantalla de cuatro zonas:

```
ARBOL     CALLE     AC
```

Por último, hay que tener en cuenta que en algunos «micros» es necesario definir DEF FNA(X) antes de la utilización de FNA(X), por eso interesa definir las funciones al principio del programa.

Ejercicios

8.9. Definir una función para el polinomio $3x^2+2x+1$.

8.10. Por medio de una función, hacer un programa que escriba los cubos de los 20 primeros números naturales.

8.11. Hacer un programa que defina una función que sume el seno y el coseno de un mismo ángulo, imprimiendo el resultado.

Subrutinas

Frecuentemente ocurre que un mismo conjunto de instrucciones hay que repetirlo varias veces a lo largo de un mismo programa.

Así, en el programa que sigue:

```
 10 REM CALCULO DE LA SUPERFICIE DE UN CIRCULO PARA DISTINTOS RADIOS
 20 LET R=1
 30 GOSUB 200
 40 LET R=2
 50 GOSUB 200
 60 LET R=3
 70 GOSUB 200
 80 STOP
200 REM SUBRUTINA
210 PRINT "S= "; 3.1415 * R ↑ 2
220 RETURN
230 END
```

las líneas 30 GOSUB 200, 50 GOSUB 200 y 70 GOSUB 200 hacen saltar el control del programa a la instrucción 200.

Al llegar a la instrucción 220 RETURN el control vuelve a la instrucción siguiente a la GOSUB.

El orden de ejecución del programa es el siguiente:

```
 10 REM CALCULO DE LA SUPERFICIE DE UN CIRCULO PARA DISTINTOS RADIOS
 20 LET R=1
 30 GOSUB 200
  |
  └──────────────→ 200 REM SUBRUTINA
                   210 PRINT "S= "; 3.1415 * R ↑ 2
  ┌─────────────── 220 RETURN
  ↓
 40 LET R=2
 50 GOSUB 200
  |
  └──────────────→ 200 REM SUBRUTINA
                   210 PRINT "S= "; 3.1415 * R ↑ 2
  ┌─────────────── 220 RETURN
  ↓
 60 LET R=3
 70 GOSUB 200
  |
  └──────────────→ 200 REM SUBRUTINA
                   210 PRINT "S= "; 3.1415 * R ↑ 2
  ┌─────────────── 220 RETURN
  ↓
 80 STOP
230 END
```

Programa: comprobación del resultado de la suma y del producto de dos números

```
 10  REM SUMAR Y MULTIPLICAR
 20  PRINT ''INTRODUCE DOS NUMEROS''
 30  INPUT A
 40  PRINT A
 50  INPUT B
 60  PRINT B
 70  LET T=A+B
 80  PRINT A; ''+''; B; ''=''
 90  INPUT R
100  GOSUB 500
110  LET T=A * B
120  PRINT A; '' * ''; B; ''=''
130  INPUT R
140  GOSUB 500
150  STOP
500  IF R=T THEN 530
510  PRINT ''ERROR, EL RESULTADO ES''; T
520  GOTO 540
530  PRINT ''EL RESULTADO''; T; ''ES CORRECTO''
540  RETURN
550  END
```

En este programa hay una subrutina entre las instrucciones 500 a 540. Se ha necesitado dos veces, al ejecutar la instrucción 100 y al ejecutar la 140. Nótese que la última instrucción de una subrutina siempre debe ser RETURN, para indicar al microordenador que debe volver a la instrucción siguiente a la que estaba cuando «saltó» a ejecutar la subrutina.

GOSUB n – RETURN

Formato: GOSUB n
RETURN

Función: GOSUB n transfiere el control a la instrucción n.
RETURN devuelve el control a la instrucción siguiente a la GOSUB n.

EJERCICIOS DE RECAPITULACION

8.1. Definir una función para las siguientes fórmulas:

a) $z=\frac{x+5}{2}$

b) $z=x^3+2x-3$

c) $z=x+LOG\ x$

d) $z=x\cdot 5^x$

8.2. En las siguientes partes de programas que contienen subrutinas existen algunos fallos y errores; indicarlos:

a)
```
 10 GOSUB 100
 .....
100 REM SUBRUTINA
 .....
1700 GOTO 30
 .....
200 RETURN
300 END
```
subrutina

b)
```
 20 GOSUB 130
 .....
130 INPUT X
 .....
180 END
```
subrutina

8.3. Elaborar un programa que genere una tabla de la función seno entre 0 y π, incrementando la variable X de 0.2 en 0.2.

8.4. Diseñar un programa que simule el lanzamiento de una moneda imprimiendo "CARA" si el resultado es cara y "CRUZ" en caso contrario.

8.5. Escribir un programa que dé el valor absoluto de un número, sin utilizar la función ABS(X).

8.6. La fórmula que permite calcular el logaritmo en base *a* de un número x conociendo el logaritmo en base *e* (logaritmo neperiano o natural) del mismo número x es:

$$\log_a x=\frac{\log_e x}{\log_e a}$$

Diseñar un programa que calcule el logaritmo decimal de una serie de números.

8.7. Las funciones cosecante, secante y cotangente vienen definidas por:

$$\text{cosec } x=\frac{1}{\text{sen } x},$$

$$\text{sec } x=\frac{1}{\cos x},$$

$$\text{ctg } x=\frac{1}{\text{tg } x}.$$

Hacer un programa que las defina, llamándolas FNC(X), FNX(X) y FNG(X), respectivamente, y dé su resultado desde x=0.2 a π incrementándose x de 0.5 en 0.5.

8.8. La parte decimal de un número positivo se obtiene restándole su parte entera; en BASIC.

```
N-INT (N)
```

Diseñar un programa que defina la función «parte decimal» y escriba el resultado para cualquier número que se introduzca.

8.9. ¿Cómo se tendría que definir la función parte decimal si el número N fuera negativo?

8.10. Escribir un programa que defina la función polinómica de tercer grado

$$F(x)=Ax^3+Bx^2+Cx+D$$

e introduzca como valores A=3, B=2, C=1 y D=0.
Evaluar esta función desde I=1 hasta 10 incrementando a x de 1 en 1.

8.11. Hacer un programa que dé la media de dos números aleatorios, definiendo la función media.

9 Gráficos

Instrucción PRINT TAB

En este apartado vamos a estudiar la posibilidad de obtener en la pantalla gráficos sencillos, susceptibles de ser dibujados por todos los «micros». Consideraremos un microordenador capaz de imprimir en 22 filas (de la 0 a la 21) y en 32 columnas (de la 0 a la 31).

La instrucción PRINT TAB (imprimir tabulando) sirve para indicar en qué columna se ha de imprimir. Así, por ejemplo,

```
PRINT TAB (8); "X"
```

significa imprimir una X en la columna 8.

Supongamos que se quiere obtener una tabla que dé en la columna 3 un número, en la 8 su cuadrado y en la 19 su seno (ángulo en radianes). El programa será:

```
 1  REM CUADRADO Y SENO DE UNA SERIE DE NUMEROS
10  INPUT "CUANTOS NUMEROS SON"; N
20  PRINT TAB (3); "X"; TAB (8); "X ↑ 2"; TAB (19); "SIN (X)"
30  FOR I = 1 TO N
40  INPUT X
50  PRINT TAB (3); X; TAB (8); X ↑ 2; TAB (19); SIN (X)
60  NEXT I
70  END
```

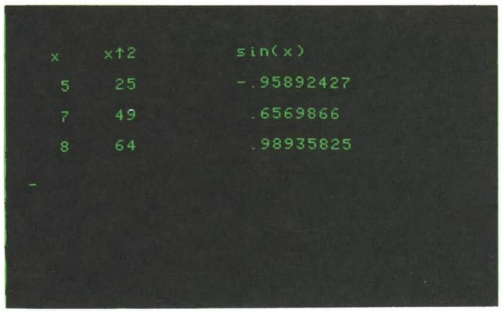
La instrucción 10 pide el número que se va a introducir y la 20 imprime X, X ↑ 2 y SIN (X) en las columnas 3, 8 y 19 respectivamente, de la fila 0. Los separadores (;) operan de la misma forma que en una instrucción PRINT sin TAB. Al no haber ningún separador después de «SIN(X)», la siguiente instrucción PRINT imprimirá en la siguiente fila.

El bucle 30 - 60 repite N veces el conjunto formado por las instrucciones 40 y 50. La 40 pide un valor X y la 50 imprime ordenadamente en las columnas 3, 8 y 19 los valores de X, X ↑ 2 y SIN(X), respectivamente.

Si al ejecutar el programa, por medio de RUN [ENTER], se introducen los datos

N = 3; X = 5; X = 7; X = 8

en la pantalla aparecerá:



Ejercicios

9.1. Elaborar un programa que produzca los mismos resultados que el que se acaba de analizar sustituyendo el bucle FOR-NEXT

9.2. Hacer un programa que escriba en la columna 10 «NOMBRE», en la 20 «PRIMER APELLIDO» y que vaya admitiendo nombres y apellidos de personas imprimiéndolos en las columnas correspondientes.

Gráficos PRINT TAB

La instrucción PRINT TAB también se puede utilizar para dibujar gráficos, aunque su manejo no siempre es sencillo.

El programa para dibujar los dos ejes coordenados sería de la forma:

```
  1  REM EJES COORDENADOS
 10  FOR I=1 TO 9
 20  PRINT TAB (15); " * "
 30  NEXT I
 40  FOR J=0 TO 31
 50  PRINT " * ";
 60  NEXT J
 70  FOR K=11 TO 21
 80  PRINT TAB (15); " * "
 90  NEXT K
100  END
```

(La instrucción 20, al no terminar en coma ni en punto y coma, hace imprimir el próximo carácter en la fila siguiente.)

(La instrucción 50, al terminar en punto y coma, hace que el siguiente carácter se escriba a continuación.)

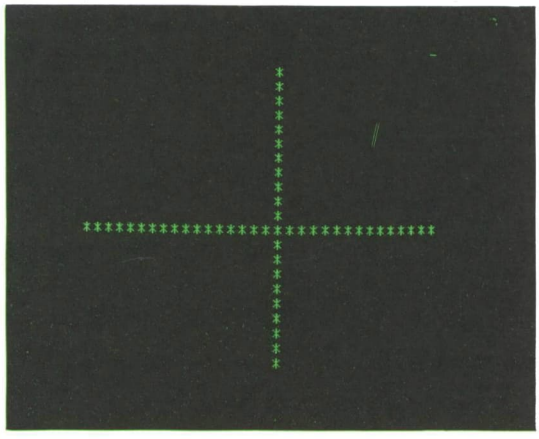
(La instrucción 80 es idéntica a la 20.)

El primer bucle 10 - 30, de variable I, imprime en la columna 15 un asterisco desde la fila 1 a la 9.

El segundo bucle 40 - 60, de variable J, llena de asteriscos toda la fila 10. (En este problema en concreto se puede suprimir TAB (J), pues J varía de 0 a 31 y la instrucción PRINT termina en punto y coma.)

Por último, el bucle 70 - 90, de variable K, continúa imprimiendo asteriscos en la columna 15, de la fila 11 a la 21.

Si se ejecuta el programa tecleando (RUN ENTER), se obtiene:



Ejercicios

9.3. Hacer un programa que dibuje un asterisco en la fila 10, columna 15. (Para llegar a la fila 10, introducir un bucle.)

9.4. Realizar un programa que llene la columna 12 de asteriscos.

9.5. Hacer un programa que imprima "A" en toda la fila 3.

9.6. Escribir un programa que dibuje una recta que vaya de (0, 0) a (21, 21).

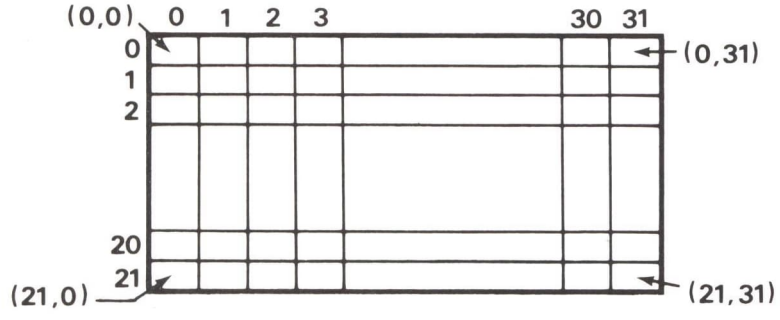
9.7. Hacer un programa que dibuje una recta de (0, 31) a (21, 0).

Instrucción PRINT AT (o PLOT)

La instrucción PRINT AT sirve para escribir en la pantalla un determinado signo en la posición dada por dos coordenadas. En algunos «micros» la instrucción equivalente se escribe directamente PLOT (*).

Aquí también vamos a considerar un microordenador con posibilidades de escribir en 22 filas y en 32 columnas.

Como el cursor o punto luminoso que sigue la escritura de los caracteres suele encontrarse inicialmente en la posición superior izquierda, a esta posición se asigna las coordenadas (0, 0), a la superior derecha (0, 31), a la inferior izquierda (21, 0) y, por último, a la inferior derecha (21, 31).



(*) No todos los «micros» poseen la instrucción PRINT AT (o PLOT).

Estudiamos ahora un programa que dibuje unos ejes de coordenadas con centro en el punto de coordenadas (10, 15).

Para dibujar el eje horizontal utilizaremos la instrucción PRINT AT 10, Y; ''+'' (en otros «micros», PLOT 10, Y; ''+'').

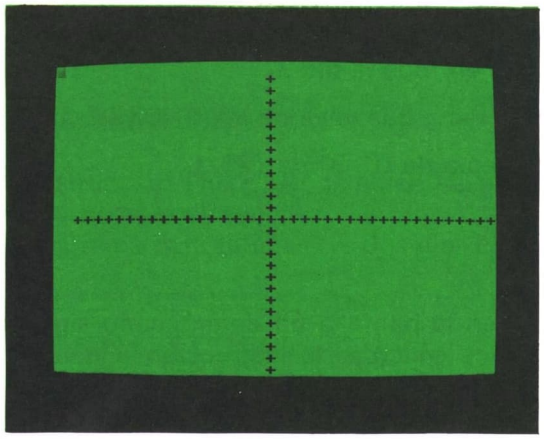
Esta instrucción imprime un signo + en el punto de coordenadas (10, Y), es decir, en la intersección de la fila 10 y de la columna Y (**). Se ha puesto una Y como segunda coordenada para que varíe de 0 a 31, gracias al bucle FOR-NEXT.

Para dibujar el eje vertical se hace variar la primera coordenada X: PRINT AT X, 15; ''+''.

El programa es

```
 1  REM EJES DE COORDENADAS
10  FOR Y=0 TO 31
20  PRINT AT 10, Y; ''+''
30  NEXT Y
40  FOR X=0 TO 21
50  PRINT AT X, 15; ''+''
60  NEXT X
70  STOP
```

que al ejecutarlo (RUN ENTER) hace aparecer en la pantalla el siguiente resultado:



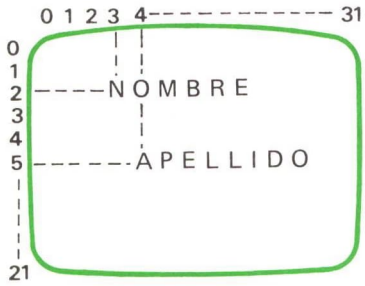
En vez de puntos se pueden imprimir guiones o cualquier otro carácter o cadena que interese, pues la instrucción PRINT AT X, Y; '' '' lo que hace es empezar a imprimir en las coordenadas (X, Y) lo que está entre comillas. Por ejemplo, el programa

```
 1  REM IMPRESION ORDENADA
10  PRINT AT 2, 3; ''NOMBRE''
20  PRINT AT 5, 4; ''APELLIDOS''
30  STOP
```

(**) En algunos microordenadores el primer número indica la columna y el segundo la fila. Así, estos «micros» con PRINT AT 10, Y; ''+'' imprimen un signo + en la columna 10 y en la fila Y.

escribe NOMBRE a partir del punto de coordenadas (2, 3) y APELLIDO desde el punto (5, 4).

Al ejecutar (RUN [ENTER]), se obtiene en pantalla:



Ejercicios

9.8. Hacer un programa que dibuje una línea recta en la fila 3.

9.9. Escribir un programa que imprima la gráfica de la función constante f (x) = 7.

9.10. Hacer un programa que dibuje las diagonales de un cuadrado de 21×21 (no considerar las columnas 22 a 31).

9.11. Hacer un programa que dibuje los ejes de coordenadas y la bisectriz del primer cuadrante, y = x.

9.12. Elaborar un programa que dibuje un rectángulo en la pantalla.

Gráficos en alta resolución

Si un «micro» es tal que puede escribir en 22 filas y en 32 columnas de la pantalla, el número total de puntos que puede localizar en dicha pantalla es 22×32=704 puntos. Con este microordenador, si se intenta dibujar una curva, por ejemplo una parábola, no aparecerá como curva continua.

Si, en cambio, el microodenador es capaz de localizar en la pantalla del orden de los 50.000 puntos o más, la curva aparecerá con saltos insignificantes y se verá como continua. Se dice que el primer microordenador escribe o dibuja en *baja resolución* y el segundo en *alta resolución*.

Para gráficos en alta resolución cada «micro» trabaja con instrucciones específicas.

En el capítulo 12, REPRESENTACION GRAFICA DE FUNCIONES, complementamos los conceptos desarrollados aquí.

EJERCICIOS DE RECAPITULACION

9.1. Hacer un programa que vaya admitiendo números y tabule la pantalla para que en la columna 10 aparezcan los números que se van introduciendo y en la columna 20 el número más pequeño introducido hasta este momento.

9.2. Elaborar un programa que imprima en la columna 5 el nombre de una persona, en la 13 su edad y en la 20 su peso. Además, mediante READ-DATA introducir los nombres, edades y pesos de seis personas.

9.3. Hacer un programa que dibuje, mediante la instrucción PRINT TAB, la letra I, con una altura de diez puntos y anchura de tres puntos, imprimiendo asteriscos.

9.4. Hacer un programa que dibuje la letra A con las mismas condiciones que en el programa anterior.

9.5. Diseñar un programa que dibuje la función seno mediante la instrucción PRINT TAB, de tal manera que la gráfica del seno aparezca en posición vertical.

9.6. El siguiente programa dibuja la función seno en una pantalla con baja resolución.

```
10  FOR I=0 TO 31
20  PRINT AT 10 INT (10 - SIN (I * PI/15), I; '' * ''
30  NEXT I
```

Modificar este programa para que dibuje el coseno.
(Observación: en la instrucción 20 aparece PI, cuyo valor es 3.14159... La mayoría de los microordenadores al leer PI asignan directamente el valor 3.14159...)

9.7. Hacer un programa que dibuje la función $F(x)=x^2$ mediante instrucciones PRINT AT.

9.8. Diseñar un programa que dibuje la función exponencial $F(x) = e^x$ utilizando la instrucción PRINT TAB.

APLICACIONES

10 Introducción a las aplicaciones

Algunas pautas para resolver problemas con microordenadores

La solución de cualquier problema en informática constituye un proceso de investigación y, como tal, no se puede establecer un conjunto de reglas fijas que siempre conduzcan a una solución correcta. Sin embargo, ciertas pautas o consejos pueden ser útiles en ocasiones. Resumiremos a continuación algunas de estas ideas que facilitan el trabajo.

1. Definición del problema

Se trata de simplificar y aislar el problema en que estamos interesados, eliminando toda la información no necesaria, para utilizar así el menor número posible de conceptos.

2. Análisis del problema

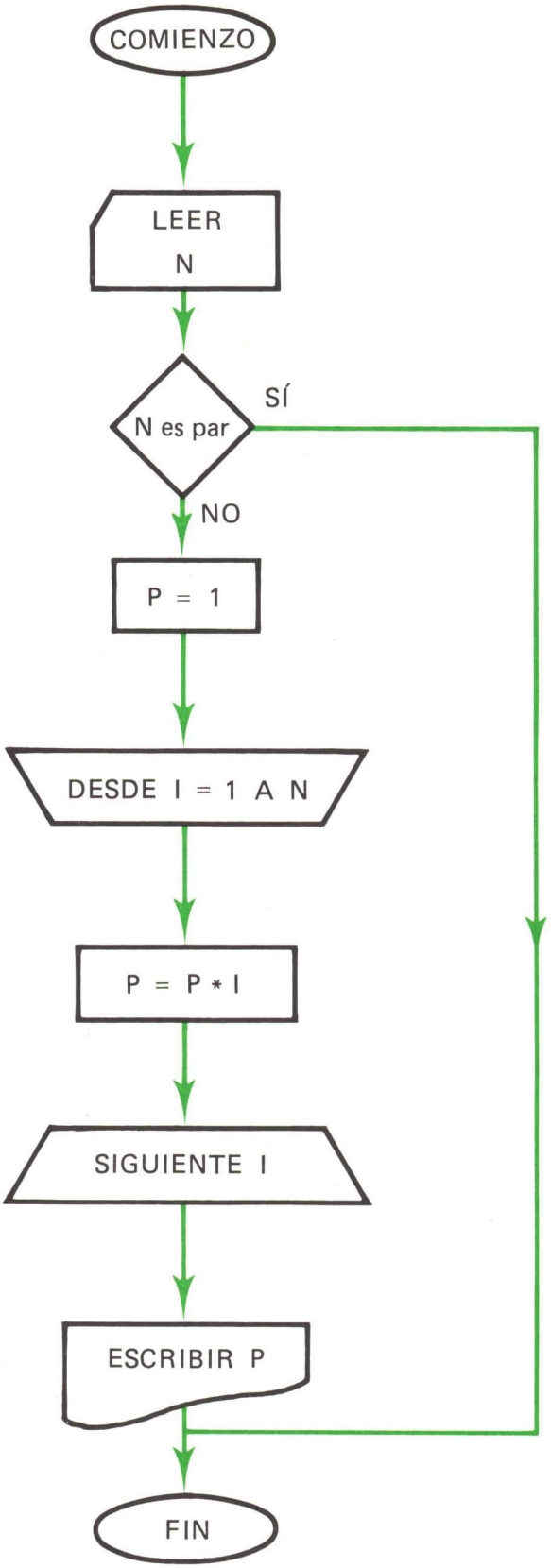
Una vez definido el problema debemos elegir una estrategia para llegar a una solución. Como se va a emplear un microordenador es necesario imponer una serie de condiciones, que vienen dadas por las instrucciones que éste es capaz de «entender» y realizar. Este proceso nos llevará así a un procedimiento, es decir, a un número finito de pasos (algoritmo), que conduzcan a la solución del problema.

3. Organigramas o diagramas de flujo

Los distintos pasos que supone el análisis de un problema se pueden indicar mediante esquemas gráficos. El conjunto de estos pequeños pasos vinculados entre sí y expresados gráficamente constituye lo que se llama *organigrama* o *diagrama de flujo.*

En un diagrama de flujo, las diferentes etapas que se van a ejecutar se representan en recuadros en los que se incluyen las instrucciones específicas. El orden en el cual las etapas se deben ejecutar se indica mediante el sentido de las flechas que unen dichos recuadros.

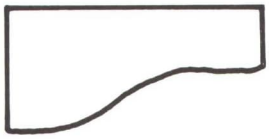
Como ejemplo, consideremos el diagrama de flujo correspondiente a un programa que lee un número N, y calcula el producto de los N primeros números naturales si N no es par y, en caso contrario, no hace nada.


COMIENZO
LEER
N
N es par
SÍ
NO
P = 1
DESDE I = 1 A N
P = P * I
SIGUIENTE I
ESCRIBIR P
FIN

Los símbolos que se han utilizado son:

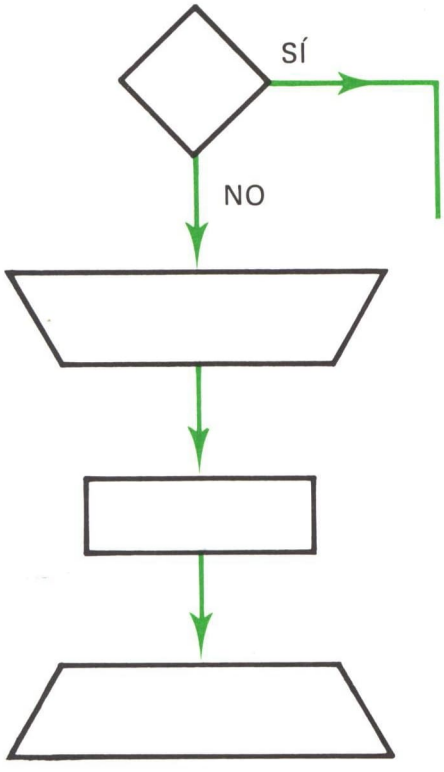
comienzo o final (END)
del programa

lectura
(INPUT)



escritura
(PRINT)

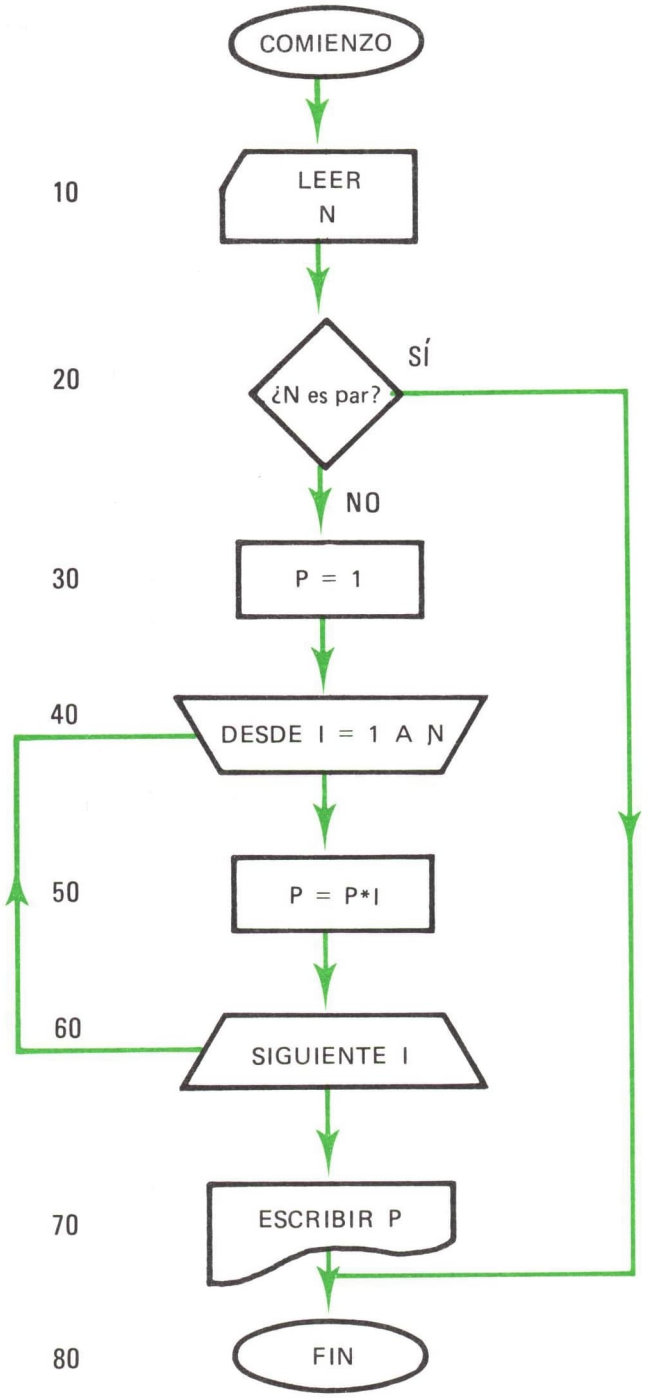
asignación
(LET)



toma de decisión
(IF... THEN...)

repetición de las instrucciones
intermedias
(FOR... TO... STEP...; NEXT...)

Como se puede observar, a cada símbolo le corresponde una instrucción en BASIC (excepto a COMIENZO), por lo que es bastante fácil pasar del organigrama al correspondiente programa:




```
10  INPUT  N
20  IF  N = 2*INT(N/2)  THEN 80
30  LET  P = 1
40  FOR  I = 1 TO  N
50  LET  P = P*I
60  NEXT  I
70  PRINT  P
80  END
```

La instrucción 20 averigua si N es par o no. Para ello se utiliza la función INT (parte entera). Por ejemplo, si mediante INPUT se introduce N=25, se tendría:

```
IF N=2 * INT(25/2) THEN 80
```

El microordenador hace estos cálculos:

2 * INT(12.5)=2 * 12=24

y a continuación compara 25 con 24 (¿N=24?). Como no se cumple la igualdad (lo que supone que N no es par), ejecuta la instrucción siguiente a la 20.

4. Elaboración de un programa

Para materializar el procedimiento, pueden tenerse en cuenta los siguientes pasos:

a) Traducir todas las informaciones que manejamos a variables y conjuntos de variables.

b) Traducir todas las manipulaciones de datos a estructuras. Es decir, subdividir el programa en posibles bucles y subrutinas.
La utilización de subrutinas facilita una concepción y presentación estructurada de los programas. Cada programa puede considerarse, entonces, como una asociación de bloques de instrucciones, constituyendo uno de ellos el programa principal y los restantes, bloques anexos (las subrutinas o subprogramas).

c) Tratar de desarrollar el programa en un conjunto de bloques lo más completos posible en sí mismos. (En este sentido es útil usar REM para identificar las distintas partes.)

5. Verificación del programa

a) Para detectar errores en un programa, podemos ejecutarlo con datos para los cuales se conocen los resultados y compararlos a continuación con las soluciones obtenidas.

b) Si un programa contiene distintas bifurcaciones, debemos utilizar datos que permitan comprobar todas las opciones.

c) La técnica de construcción de un programa en diversos módulos facilita su verificación, ya que usando instrucciones de entrada y salida adicionales (que luego se eliminarán) permiten verificar los procesos intermedios.

d) Una técnica de depuración útil (sobre todo en programas cortos) es seguir los pasos del programa con lápiz y papel, simulando el trabajo del microordenador.

Objetivos de las aplicaciones

En las aplicaciones (capítulos 11 y siguientes) que siguen se utilizan estas ideas. En dichas aplicaciones se pretende dar una muestra de la gran cantidad de problemas que pueden resolverse con un microordenador. Al mismo tiempo, constituyen verdaderos ejercicios, los cuales permitirán conseguir una mayor autonomía a la hora de programar.

Algunas de estas aplicaciones, sobre todo las más sencillas, el alumno puede elegirlas como ejercicios y, para comprobar si están bien programadas, conviene que las ejecute en el microordenador y las compare con el programa que se da en este libro.

Ejercicios

10.1. Expresar en un organigrama el proceso que hay que seguir para sumar los N primeros números naturales.

10.2. Obtener un organigrama en el que se indique el cálculo de las raíces reales de una ecuación de segundo grado.

10.3. Expresar en un organigrama el proceso que hay que seguir para averiguar si un número es primo o no.

11 Matemáticas

Resolución de la ecuación de segundo grado

Una ecuación de segundo grado $Ax^2+Bx+C=0$ tiene siempre dos soluciones, x_1 y x_2, dentro del cuerpo $\mathbb{C}$ de los números complejos. En el programa siguiente se van a obtener estas dos soluciones, indicando si son reales o complejas. Primeramente se introducen como datos los coeficientes A, B y C.

Programa:

```
 1  REM RESOLUCION DE UNA ECUACION DE SEGUNDO GRADO
10  READ A, B, C
20  LET D=B ↑ 2-4 * A * C
30  IF D < 0 THEN 80
```

En la instrucción 10 se leen los coeficientes A, B y C de la ecuación. La 20 asigna a D el valor del discriminante B^2-4AC. Por último, la instrucción 30 pregunta si el discriminante es negativo y, en tal caso, envía el control a la instrucción 80, para calcular las soluciones complejas. Si no se verifica que $D<0$, las raíces son reales y su cálculo comienza en la instrucción 40.

```
40  LET X1=(-B+SQR(D)) / (2 * A)
50  LET X2= (-B-SQR(D)) / (2 * A)
60  PRINT ''LAS SOLUCIONES SON REALES Y SUS''
65  PRINT ''VALORES SON: ''; X1; '' Y ''; X2
70  STOP
```

En las instrucciones 40 y 50 se calculan las dos soluciones reales X1 y X2 (que son iguales cuando D=0). En la 60 se escriben las soluciones y en la 70 se para el programa.

```
80  LET PR=-B / (2 * A)
90  LET PY=SQR(-D) / (2 * A)
100 PRINT ''LAS SOLUCIONES SON COMPLEJAS Y''
105 PRINT ''SUS VALORES SON: ''; PR; ''+''; PI; ''I''; ''Y''; PR; ''-''; PY; ''I''
```

La instrucción 80 calcula la parte real PR y la 90 la parte imaginaria PY. Como los dos números complejos son conjugados, basta con cambiar el signo de la parte imaginaria para obtener las soluciones en la instrucción 100.

Por contener el programa la instrucción 10 READ A, B, C, tiene que haber una instrucción DATA con tres datos. Por eso tendremos que añadir, por ejemplo, la instrucción:

```
110  DATA 1, -4, 3
```

Si se ejecuta el programa tecleando RUN [ENTER] se obtiene este resultado en una pantalla con 32 columnas:

```
LAS SOLUCIONES SON REALES Y SUS
VALORES SON: 3 Y 1
```

Si se cambian los datos de la instrucción 110 por 1, −2, 10 se obtienen soluciones complejas:

```
110  DATA 1, -2, 10
```

Al ejecutar el programa nuevamente (RUN [ENTER]) se obtiene como resultado:

```
LAS SOLUCIONES SON COMPLEJAS Y
SUS VALORES SON: 1+3I Y 1-3I.
```

Ejercicios

11.1. En el programa anterior sólo se dan las dos soluciones reales o las dos complejas, pero no se especifica el caso en que las dos soluciones reales sean coincidentes. Modificar el programa anterior de tal forma que se dé un informe por medio de una instrucción PRINT, en el caso de que las dos soluciones reales sean idénticas.

11.2. Sustituir en el programa de la ecuación de segundo grado las instrucciones READ y DATA por INPUT.

Aprendiendo a multiplicar

Se generan dos números al azar de una cifra y se pregunta cuál es su producto. Después de comparar la contestación con el resultado correcto, se anotan los aciertos y, en caso de error, se dan dos nuevas ocasiones. Se proponen 10 números utilizando un bucle (I=1 a 10).

Programa

```
  1  REM MULTIPLICAR
 10  LET C=0
 20  LET D=0
 30  FOR I=1 TO 10
 40  LET A=INT(RND(1) * 9+1)
 50  LET B=INT(RND(1) * 9+1)
 60  LET P=A * B
 70  PRINT A; '' * ''; B; ''=''
 80  INPUT X
 90  IF X=P THEN 150
100  LET D=D+1
110  PRINT ''LO SIENTO: TE EQUIVOCASTE''
120  IF D > 2 THEN 160
130  PRINT ''INTENTALO DE NUEVO''
140  GOTO 70
150  LET C=C+1
160  PRINT A; '' * ''; B; ''=''; P
170  NEXT I
180  PRINT ''EL NUMERO DE ACIERTOS ES''; C
190  END
```

Con las instrucciones 10 y 20 se inicializan los dos contadores C y D.

En las instrucciones 40, 50 y 60 se generan los números de una cifra A y B al azar, multiplicándolos a continuación y almacenando el resultado en P.

Se pide una contestación mediante la instrucción 80 y se compara en 90 con la solución correcta. Si la contestación es acertada, se suma 1 al contador C, en la 150, y se escribe el resultado correcto. Si es errónea, se suma 1 al contador D, en la 100; según el valor de éste, se dan nuevas oportunidades (desde D=0 hasta D=2). El proceso se repite 10 veces.

Finalmente, en 180 se imprime el contenido final del contador C, el cual da el número de aciertos.

Ejercicios

11.3. Escribir un programa semejante al anterior que permita ejercitarse en el producto de números de una cifra por otros de dos.

11.4. Escribir un programa semejante al anterior que sirva para ejercitarse en la suma de números de dos cifras.

Anualidades de amortización

Las anualidades de amortización son los pagos que hay que realizar anualmente a un Banco o a una entidad financiera para amortizar un préstamo. La fórmula es:

$$A=\frac{C.I.\ (1+I)^T}{(1+I)^T-1}$$

donde:

C es el capital prestado por el Banco,
I es el tanto por uno anual,
T es el tiempo en años,
A es las anualidades que hay que pagar para amortizar el préstamo C.

El siguiente programa calcula el valor de A para sucesivos valores de C, I y A hasta que se le informe al microordenador la no existencia de más datos.

Programa

```
 1  REM ANUALIDADES DE AMORTIZACION
10  INPUT ''HAY MAS DATOS'', X$
20  IF X$=''NO'' THEN 100
30  INPUT ''CAPITAL'', C
40  INPUT ''INTERES'', I
50  INPUT ''TIEMPO'', T
```

En la instrucción 10 se pregunta si hay más datos; si no los hay, la instrucción 20 envía el control al final del programa (instrucción 100). Si hay más datos, en las instrucciones 30, 40 y 50 se introducen los valores de C, I y T.

```
60  LET Z=(1+I) ↑ T
70  LET A=(C * I * Z) / (Z-1)
80  PRINT ''LAS ANUALIDADES SON'', A
90  GOTO 10
100 END
```

En las instrucciones 60 y 70 se calcula el valor de A. La 80 imprime el valor de la anualidad A y la 90 envía el control a la instrucción 10.

Si, por ejemplo, se quieren calcular las anualidades para los datos

C=1.000.000	y	C=2.000.000
I=0,15		I=0,10
T=5		T=10

se teclea RUN [ENTER] y aparece en pantalla:

```
HAY MAS DATOS ?
```

Se escribe

```
SI
```

y a continuación se introducen los primeros valores de C, I y T, apareciendo en pantalla el resultado.

LAS ANUALIDADES SON: 298315.55

Vuelve a aparecer en pantalla el mensaje

HAY MAS DATOS ?

Se imprime

SI

y se introducen los segundos valores de C, I y T, obteniendo como resultado

LAS ANUALIDADES SON: 325400.79

Por último aparece en la pantalla

HAY MAS DATOS ?

Se imprime

NO

y termina la ejecución.

Ejercicios

11.5. Sabiendo que la fórmula que da las anualidades de capitalización es:

$$A = \frac{C.I.}{(1+I)\ [(1+I)^T - 1]}$$

escribir un programa que calcule estas anualidades.

11.6. Modificar el programa de las anualidades de amortización para el caso en el que el interés nominal venga dado en tanto por ciento anual.

11.7. Modificar el programa para obtener una tabla para capital C e interés nominal I, fijos, con el tiempo T variable de 1 a 20 años.

Suma de matrices

Una matriz es una tabla numérica distribuida en filas y columnas. Por ejemplo, la siguiente distribución de números

	1.ª columna	*2.ª columna*	*3.ª columna*
1.ª fila	2	3	4
2.ª fila	1	0	−2

es una matriz de 2 filas y 3 columnas. Si a la matriz anterior se le llama A, el elemento que ocupa la primera fila y la segunda columna, que es 3, suele indicarse a_{12}, y en trabajos de programación, A (1, 2). El primer número indica la fila y el segundo la columna. Luego los elementos de la matriz A son:

A (1, 1)=2 A (1, 2)=3 A (1, 3)=4
A (2, 1)=1 A (2, 2)=0 A (2, 3)=−2

La matriz A se dice que es de dimensión 2×3, pues tiene 2 filas y 3 columnas. Para sumar dos matrices A y B, ambas tienen que ser de la misma dimensión (igual número de filas e igual número de columnas). La matriz suma se obtiene sumando los términos que ocupan la misma posición. Ejemplo:

$$\begin{pmatrix} 2 & 3 & 4 \\ 1 & 0 & -2 \end{pmatrix} + \begin{pmatrix} 6 & 3 & 1 \\ -4 & 2 & 5 \end{pmatrix} = \begin{pmatrix} 8 & 6 & 5 \\ -3 & 2 & 3 \end{pmatrix}$$

Programa

En un programa que sume matrices, primeramente se debe indicar la dimensión de las mismas, M y N. A continuación se definen las matrices A(M, N); B(M, N); S(M, N), siendo A y B las matrices sumandos (datos) y S la matriz suma.

```
 1  REM SUMA DE MATRICES
10  INPUT "FILAS", M
20  INPUT "COLUMNAS", N
30  DIM A (M, N)
40  DIM B (M, N)
50  DIM S (M, N)
```

En las instrucciones INPUT se introducen las dimensiones (M=número de filas y N=número de columnas). En las instrucciones 30, 40 y 50 se dimensionan las matrices A, B y S.

```
60  PRINT "MATRIZ A"
70  FOR I=1 TO M
80  PRINT "FILA"; I
90  FOR J=1 TO N
100 PRINT "COLUMNA"; J
110 INPUT A (I, J)
120 NEXT J
130 NEXT I
140 CLS
```

Mediante el bucle 70 - 130 se hace variar la fila y mediante el 90 - 120, la columna. La instrucción 110 va pidiendo los distintos valores de la matriz A. La 140 borra la pantalla.

```
150 PRINT "MATRIZ B"
160 FOR I=1 TO M
170 PRINT "FILA"; I
180 FOR J=1 TO N
190 PRINT "COLUMNA"; J
200 INPUT B (I, J)
210 NEXT J
220 NEXT I
230 CLS
```

Este bloque repite el mismo proceso que el anterior, sólo que ahora se introducen los valores de la matriz B.

```
240  FOR I=1 TO M
250  FOR J=1 TO N
260  LET S (I, J)=A (I, J)+B (I, J)
270  NEXT J
280  NEXT I
```

Por medio de los bucles 240 - 280 (que hace variar las filas) y 250 - 270 (que hace variar las columnas) se van realizando, en la instrucción 260, las sumas de los elementos que ocupan la misma posición en A y B, almacenándose el resultado en S(I, J).

```
290  PRINT ''LA MATRIZ SUMA ES''
300  FOR I=1 TO M
310  PRINT ''FILA''; I
320  FOR J=1 TO N
330  PRINT S (I, J)
340  NEXT J
350  NEXT I
360  END
```

En este grupo de instrucciones se da la matriz suma S por filas, terminando el programa. Si se ejecuta el programa para M=2 y N=3, siendo las matrices

$$A=\begin{pmatrix} 2 & 3 & 4 \\ 1 & 0 & -2 \end{pmatrix} \qquad B=\begin{pmatrix} 6 & 3 & 1 \\ -4 & 2 & 5 \end{pmatrix}$$

se obtiene en pantalla

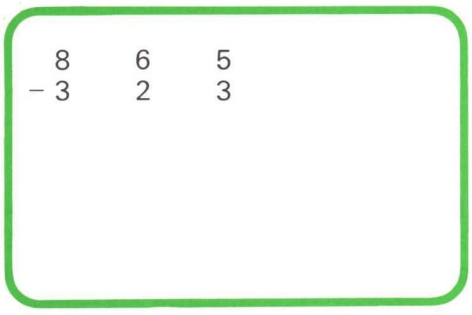
```
FILA 1
 8
 6
 5

FILA 2
-3
 2
 3
```

En consecuencia, la matriz suma es:

$$S=\begin{pmatrix} 8 & 6 & 5 \\ -3 & 2 & 3 \end{pmatrix}$$

Ejercicio

11.8. Modificar la salida del programa anterior para obtener en pantalla



utilizando una instrucción PRINT TAB.

11.9. Hacer un programa que realice el cálculo de la suma de dos matrices de tal manera que introduzca los datos por medio de READ y DATA.

11.10. Escribir un programa que reste matrices.

11.11. Para multiplicar un número por una matriz se multiplica por cada elemento de dicha matriz. Ejemplo:

$$5\begin{pmatrix} 2 & 3 & 4 \\ 1 & 0 & -2 \end{pmatrix} = \begin{pmatrix} 10 & 15 & 20 \\ 5 & 0 & -10 \end{pmatrix}$$

Hacer un programa que multiplique números por matrices.

Producto de matrices

Para poder multiplicar dos matrices A y B es necesario que el número de columnas de A coincida con el número de filas de B.

Si la dimensión de A es M×N, la dimensión de B tiene que ser N×P; es decir, el número de columnas de la primera matriz tiene que coincidir con el número de filas de la segunda.

La matriz producto R=A×B tiene por elementos:

R(1, 1) = suma de los productos de los términos de la 1.ª fila de A por los de la 1.ª columna de B.

R(1, 2) = suma de los productos de los términos de la 1.ª fila de A por los de la 2.ª columna de B.

. . . . . .

R(1, P) = suma de los productos de los términos de la 1.ª fila de A, por los de la P-ésima columna de B.

R(2, 1) = suma de los productos de los términos de la 2.ª fila de A por los de la 1.ª columna de B.

R(2, 2) = suma de los productos de los términos de la 2.ª fila de A por los de la 2.ª columna de B.

. . . . . .

Ejemplo:

Sean

$$A=\begin{pmatrix}1 & 2 & 3\\ 4 & 0 & 5\end{pmatrix} \text{ y } \qquad B=\begin{pmatrix}6 & 3\\ 1 & 0\\ 2 & 5\end{pmatrix}$$

$$R=A\times B=\begin{pmatrix}1 & 2 & 3\\ 4 & 0 & 5\end{pmatrix}\begin{pmatrix}6 & 3\\ 1 & 0\\ 2 & 5\end{pmatrix}=$$

$$=\begin{pmatrix}1*6+2*1+3*2 & 1*3+2*0+3*5\\ 4*6+0*1+5*2 & 4*3+0*0+5*5\end{pmatrix}=\begin{pmatrix}14 & 18\\ 34 & 37\end{pmatrix}$$

Al ser la dimensión de A, 2×3 y la de B, 3×2, la dimensión de P es 2×2 (el primer 2 corresponde al número de filas de A y el segundo 2 al número de columnas de B).

Para diseñar un programa que efectúe el producto de dos matrices, se tendrán que leer sus dimensiones, dimensionar las matrices A, B y R e introducir los elementos de las matrices A y B.

Programa

```
 1  REM PRODUCTO DE MATRICES
10  INPUT ''FILAS DE LA PRIMERA MATRIZ'', M
20  INPUT ''COLUMNAS DE LA PRIMERA MATRIZ O FILAS DE LA SEGUNDA MATRIZ'', N
30  INPUT ''COLUMNAS DE LA SEGUNDA MATRIZ'', P
40  DIM A (M, N)
50  DIM B (N, P)
60  DIM R (M, P)
```

En las instrucciones 10, 20 y 30 se introducen las dimensiones de las dos matrices, teniendo en cuenta que el número de columnas de la primera es igual al número de filas de la segunda. Las instrucciones 40, 50 y 60 dimensionan las matrices A, B y R.

```
 70  PRINT ''MATRIZ A''
 80  FOR I= 1 TO M
 90  PRINT ''FILA''; I
100  FOR J= 1 TO N
110  PRINT ''COLUMNA''; J
120  INPUT A(I, J)
130  NEXT J
140  NEXT I
150  CLS
```

Este bloque sirve para introducir los elementos de la matriz A, por filas. Se utilizan dos bucles: el de variable I para variar las filas y el interno, de variable J, para variar las columnas (la fila I fija). Por último, la instrucción 150 borra la pantalla.

```
160  PRINT ''MATRIZ B''
170  FOR I= 1 TO N
180  PRINT ''FILA''; I
190  FOR J= 1 TO P
200  PRINT ''COLUMNA''; J
210  INPUT B(I, J)
220  NEXT J
230  NEXT I
240  CLS
```

Este bloque es análogo al anterior; solamente se diferencia en que introduce los elementos de la matriz B.

```
250  FOR I= 1 TO M
260  FOR J= 1 TO P
270  FOR K= 1 TO N
280  LET R(I, J)=R(I, J) + A(I, K) * B(K, J)
290  NEXT K
300  NEXT J
310  NEXT I
```

El bucle 250-310, de variable I, va fijando la fila de R y la de A.

El bucle 260-300, de variable J, fija la columna de R y la de B.

El bucle 270-290, de variable K, calcula el valor del elemento R(I, J) por medio de la instrucción 280.

Por tanto, en este bloque, se calculan los distintos elementos de la matriz producto R.

```
320  PRINT ''LA MATRIZ PRODUCTO ES''
330  FOR I= 1 TO M
340  PRINT ''FILA''; I
250  FOR J= 1 TO P
360  PRINT R(I, J)
370  NEXT J
280  NEXT I
390  END
```

Este grupo de instrucciones imprime los valores de las distintas filas de la matriz producto R. Con la instrucción 390 termina el programa.

Si se ejecuta el programa dando como datos:

$$M=2 \qquad N=3 \qquad P=2$$

$$A=\begin{pmatrix} 1 & 2 & 3 \\ 4 & 0 & 5 \end{pmatrix} \qquad B=\begin{pmatrix} 6 & 3 \\ 1 & 0 \\ 2 & 5 \end{pmatrix}$$

el resultado que se obtiene en la pantalla es:

```
FILA 1
14
18
FILA 2
34
37
```

es decir:

$$R=\begin{pmatrix}14 & 18\\ 34 & 37\end{pmatrix}$$

Ejercicios

11.12 Utilizando el programa ejemplo, calcular el producto de las matrices:

$$A=\begin{pmatrix}4 & -2 & 1 & 4\\ 5 & -3 & 4 & 1\\ 6 & 2 & 3 & 0\end{pmatrix} \qquad B=\begin{pmatrix}6 & 4\\ 1 & 1\\ 0 & 3\\ 2 & 2\end{pmatrix}$$

11.13. Modificar el programa anterior para que dé los resultados por columnas.

11.14. Modificar el programa anterior introduciendo los datos por medio de las instrucciones READ y DATA.

Resolución de sistemas de ecuaciones lineales

Vamos a analizar el caso en el que coinciden el número de ecuaciones y el número de incógnitas. Un sistema de este tipo tiene una única solución o infinitas. Sea, por ejemplo, el sistema de tres ecuaciones y tres incógnitas:

$$\left.\begin{aligned}2X+4Y+3Z&=1\\ X+2Y+2Z&=0\\ 2X-Y-Z&=4\end{aligned}\right\} \quad \text{genéricamente} \quad \left.\begin{aligned}A(1,1)*X+A(1,2)*Y+A(1,3)*Z&=B(1)\\ A(2,1)*X+A(2,2)*Y+A(2,3)*Z&=B(2)\\ A(3,1)*X+A(3,2)*Y+A(3,3)*Z&=B(3)\end{aligned}\right\}$$

Se comprueba si $A(1,1)=0$; en este ejemplo se cumple, pues $A(1,1)=2$.

Entonces se divide la primera ecuación por $A(1,1)=2$, obteniéndose un sistema equivalente:

$$\left.\begin{aligned}x+2y+\frac{3}{2}z&=\frac{1}{2}\\ x+2y+2z&=0\\ 2x-y-z&=4\end{aligned}\right\}$$

Se elimina la incógnita x de la 2.ª y 3.ª ecuación, sumándolas la primera multiplicada por $-A(2,1)=-1$ y por $-A(3,1)=-2$, respectivamente.

$$(-1)\qquad \begin{array}{r} -\frac{3}{2}z=-\frac{1}{2} \\ +\ 2z=0 \\ \hline \frac{1}{2}z=-\frac{1}{2} \end{array} \qquad (-2)\qquad \begin{array}{r} -4y-3z=-1 \\ -\ y-\ z=\ 4 \\ \hline -5y-4z=\ 3 \end{array}$$

Se obtiene el sistema equivalente:

$$\left.\begin{array}{r} x+2y+\frac{3}{2}z=\frac{1}{2} \\ \frac{1}{2}z=-\frac{1}{2} \\ -5y-4z=3 \end{array}\right\}$$

Se comprueba si $A(2,2) \neq 0$, relación que no se cumple, pues $A(2,2) = 0$ en el último sistema. En este caso se pregunta si $A(3,2) \neq 0$, relación que sí se cumple, pues $A(3,2) = -5$ intercambiándose las ecuaciones 2.ª y 3.ª, obteniéndose el sistema equivalente:

$$\left.\begin{array}{r} x+2y+\frac{3}{2}z=\frac{1}{2} \\ -5y-4z=3 \\ \frac{1}{2}z=-\frac{1}{2} \end{array}\right\}$$

Una vez ordenado el sistema de esta forma, su resolución se obtiene en forma inmediata el valor de Z, a continuación el Y, y finalmente el de X.

$$\left.\begin{array}{r} x+2y+\frac{3}{2}z=\frac{1}{2} \\ -5y-4=3 \\ \boxed{z=-1} \end{array}\right\} \Rightarrow \left.\begin{array}{r} x+2y-\frac{3}{2}=\frac{1}{2} \\ -5y+4=3 \\ z=-1 \end{array}\right\} \Rightarrow$$

$$\Rightarrow \left.\begin{array}{r} x+2y=2 \\ \boxed{y=\frac{1}{5}} \\ z=-1 \end{array}\right\} \Rightarrow \left.\begin{array}{r} x+\frac{2}{5}=2 \\ y=\frac{1}{5} \\ z=-1 \end{array}\right\} \Rightarrow \boxed{\begin{array}{l} x=\frac{8}{5} \\ y=\frac{1}{5} \\ z=-1 \end{array}}$$

Que es la única solución del sistema planteado en este ejemplo.

Este método de resolución se deriva del *método de Gauss.*

En el diseño de un programa para resolver sistemas de ecuaciones lineales hay que hacer intervenir los siguientes elementos:

- Número de ecuaciones=número de incógnitas=N.
- Matriz de coeficientes de las variables A, de dimensión M×N.
- Matriz de términos independientes B, de dimensión N.
- Matriz de incógnitas X, de dimensión N.

Programa

```
 1  REM RESOLUCION DE SISTEMAS
10  INPUT ''ECUACIONES O INCOGNITAS'', N
20  DIM A(N,N)
30  DIM B(N)
40  DIM X(N)
```

En la instrucción 10 se introduce el número N de ecuaciones o incógnitas. En las siguientes se dimensionan las matrices: A, de coeficientes; B, de términos independientes, y X, de incógnitas.

```
50  FOR I=1 TO N
60  PRINT ''ECUACION''; I
70  FOR J=1 TO N
80  PRINT ''COEFICIENTE'', J
90  INPUT A(I,J)
100 NEXT J
110 PRINT ''TERMINO INDEPENDIENTE''
120 INPUT B(I)
130 CLS
140 NEXT I
```

Este bloque permite introducir los coeficientes y el término independiente, ecuación a ecuación. La variable de bucle I indica la ecuación, y la variable J, el coeficiente. La instrucción 130 CLS borra la pantalla cada vez que se introducen todos los datos de una ecuación.

```
150 FOR I=1 TO N
160 FOR J=1 TO N
170 IF A(J, I) < > 0 THEN 210
180 NEXT J
190 PRINT ''HAY INFINITAS SOLUCIONES''
200 STOP
```

En la instrucción 150 comienza el proceso. En el bucle de variable J, en la instrucción 170, se pregunta si los coeficientes iniciales de las ecuaciones son distintas de cero. Si no hay ninguno distinto de cero, es decir, si todos son nulos, la instrucción 190 imprime «HAY INFINITAS SOLUCIONES» y el proceso se para en la instrucción 200.

Si hay algún coeficiente distinto de cero, la instrucción 170 envía el control a la 210 para seguir el proceso:

```
210 FOR K=I TO N
220 LET Z=A(I, K)
230 LET A(I, K)=A(J, K)
240 LET A(J, K)=Z
250 NEXT K
260 LET Z=B(I)
270 LET B(I)=B(J)
280 LET B(J)=Z
```

En este bloque de instrucciones se intercambian las ecuaciones I y J, pues la ecuación J tiene primer coeficiente distinto de cero y la I igual a cero.

```
285  IF I=N THEN 400
290  FOR J=I+1 TO N
300  LET A(I, J)=A(I, J) / A(I, I)
310  NEXT J
320  LET B(I)=B(I) / A(I, I)
```

Se dividen los coeficientes y el término independiente de la ecuación I por A(I, I).

```
330  FOR J=I+1 TO N
340  FOR K=I+1 TO N
350  LET A(J, K)=A(J, K)-A(J, I) * A(I, K)
360  NEXT K
370  LET B(J)=B(J)-A(J, I) * B(I)
380  NEXT J
390  NEXT I
```

La variable de bucle J indica que en las ecuaciones siguientes a la I se elimine el término de coeficiente A(J, I), es decir, se haga cero A(J, I), por lo que cambian de valor el resto de los coeficientes y el término independiente, según las instrucciones 350 y 370, respectivamente.

En la instrucción 390 termina el bucle iniciado en la instrucción 150.

```
400  LET X(N)=B(N) / A(N, N)
410  FOR I=N-1 TO 1 STEP -1
420  FOR J=I TO 1 STEP -1
430  LET B(J)=B(J)-A(J, I+1) * X(I+1)
440  NEXT J
450  LET X(I)=B(I)
460  NEXT I
```

En este bloque de instrucciones se van calculando los valores solución X(N), X(N−1), ..., X(2), X(1), en este orden, y sustituyéndose en todas las ecuaciones anteriores, calculando a su vez los nuevos términos independientes.

```
470  PRINT ''LA SOLUCION ES''
480  FOR I=1 TO N
490  PRINT ''X (''; I; '')=''; X(I)
500  NEXT I
510  END
```

El bucle de variable I imprime los N valores de la solución X. Si al ejecutar el programa se introducen los datos:

$$N=3 \qquad A=\begin{pmatrix} 2 & 4 & 3 \\ 1 & 2 & 2 \\ 2 & -1 & -1 \end{pmatrix} \text{ y } B=\begin{pmatrix} 1 \\ 0 \\ 4 \end{pmatrix}$$

El resultado que se obtiene en la pantalla es:

```
LA SOLUCION ES
X(1)=1.6
X(2)=0.2
X(3)=-1
```

Ejercicios

11.15. Poner delante del bucle 160 - 180 una instrucción IF para el caso en que I=J, modificando dicho bucle en la forma J=I+1. De esta forma se consigue que no se ejecute el bloque 210 - 280 en el caso de ser A(I, I)≠0.

11.16. Eliminar la matriz X, sustituyendo en el programa X(I) por A(I, I). Así se consigue ahorrar memoria.

11.17. El método utilizado para escribir este programa es derivado del de Gauss. Si se quiere mejorar, unificando las instrucciones de cálculo de soluciones (400 - 460) con las anteriores, ver el método de Gauss-Jordan.

Operaciones con vectores

Vamos a estudiar un programa que permita, conociendo las componentes de dos vectores, resolver las siguientes operaciones entre dichos vectores:

a) suma
b) resta
c) producto escalar
d) producto vectorial

teniendo en cuenta que si el vector $\vec{A}$ tiene como componentes (X1, Y1, Z1) y el vector $\vec{B}$ (X2, Y2, Z2), los resultados de estas operaciones se obtienen a partir de las expresiones:

$\vec{A}+\vec{B}=(X1+X2,\ Y1+Y2,\ Z1+Z2)$
$\vec{A}-\vec{B}=(X1-X2,\ Y1-Y2,\ Z1-Z2)$
$\vec{A}\cdot\vec{B}=(X1\cdot X2+Y1\cdot Y2+Z1\cdot Z2)$ (producto escalar)
$\vec{A}\cdot\vec{B}=(Y1\ Z2-Z1\ Y2,\ Z1\ X2-X1\ Z2,\ X1\ Y2-Y1\ X2)$ (producto vectorial)

Programa

```
 1  REM OPERACIONES CON VECTORES
10  PRINT "COORDENADAS VECTOR A"
20  INPUT X1, Y1, Z1
30  PRINT "COORDENADAS DEL VECTOR B"
40  INPUT X2, Y2, Z2
50  PRINT
60  PRINT "A+B="; X1+X2; ","; Y1+Y2; ","; Z1+Z2
70  PRINT "A-B="; XI-X2; ","; Y1-Y2; ","; Z1-Z2
80  PRINT "A · B="; X1 * X2+Y1 * Y2+Z1 * Z2
90  PRINT "A * B="; Y1 * Z2-Z1 * Y2; ","; Z1 * X2-X1 * Z2; ","; X1 * Y2-Y1 * X2
```

En estas instrucciones, después de introducir las coordenadas de los vectores (instrucciones 10 a 40), se obtienen e imprimen los resultados de las distintas operaciones (instrucciones 60 a 90).

```
100 PRINT "MAS DATOS (1=SI, 0=NO)"
110 INPUT X
120 IF X=1 THEN 10
130 END
```

En esta última parte se pregunta si se introducirán nuevos datos y, en caso afirmativo, se inicia el proceso de nuevo. En caso contrario, se termina el programa.

Ejercicios

11.18. Escribir un programa que realice la operación $A \cdot (B+C)$ y que permita comprobar si el producto escalar tiene la propiedad distributiva $(A \cdot (B+C)=A \cdot B+A \cdot C)$.

11.19. Hacer un programa que permita obtener el producto vectorial de tres vectores A, B, C.

Módulo y cosenos directores de dos vectores

Dadas las componentes de dos vectores en el espacio, el programa siguiente calculará:
a) el módulo de cada uno de ellos,
b) los cosenos de los ángulos que forman con los ejes X, Y, Z (cosenos directores),
c) El coseno del ángulo entre los dos vectores.

El programa averiguará, además, si los vectores son paralelos o perpendiculares.

Así, si X(1), Y(1) y Z(1) son las coordenadas del primer vector y X(2), Y(2), Z(2), las coordenadas del segundo vector, sus módulos respectivos M(1) y M(2) serán M(1)=SQR (X(1) ↑ 2+Y(1) ↑ 2+Z(1) ↑ 2) y M(2)=SQR (X(2) ↑ 2+Y(2) ↑ 2+Z(2) ↑ 2).

Los cosenos del ángulo que forman con los ejes son:

X(1) / M(1)	Y(1) / M(1)	Z(1) / M(1)
X(2) / M(2)	Y(2) / M(2)	Z(2) / M(2)

y el coseno entre los dos vectores se obtiene dividiendo el producto escalar de los dos vectores por el producto de sus módulos:

$$(X(1) * X(2)+Y(1) * Y(2)+Z(1) * Z(2)) / (M(1) * M(2))$$

Los vectores son perpendiculares si este coseno es cero, y son paralelos si su valor es 1 ó −1.

Sin embargo, debido al sistema operativo de muchos microordenadores, en el caso de que los vectores sean paralelos no se alcanzarán los valores 1 ó −1, sino los muy próximos 0.999999999 ó −0.999999999, cuestión ésta que hay que tener en cuenta al diseñar el programa.

Programa

```
  1  REM VECTORES
 10  DIM X(2)
 15  DIM Y(2)
 20  DIM Z(2)
 25  DIM M(2)
 30  PRINT ''COORDENADAS DEL VECTOR 1''
 40  INPUT X(1)
 50  INPUT Y(1)
 50  INPUT Z(1)
 55  PRINT ''COORDENADAS DEL VECTOR 2''
 60  INPUT X(2)
 65  INPUT Y(2)
 70  INPUT Z(2)
```

Hasta aquí se han definido las variables e introducido las coordenadas de cada vector.

```
 80  PRINT
 90  FOR I=1 TO 2
100  M(I)=SQR (X(I) ↑ 2+Y(I) ↑ 2+Z(I) ↑ 2)
110  IF M(I)=0 THEN 200
115  PRINT ''VECTOR''; I
120  PRINT ''MODULO''; M(I)
130  LET J=X(I) / M(I)
140  PRINT ''COSENO DEL ANGULO CON EL EJE X''; J
150  LET J=Y(I) / M(I)
160  PRINT ''COSENO DEL ANGULO CON EL EJE Y''; J
170  LET J=Z(I) / M(I)
180  PRINT ''COSENO DEL ANGULO CON EL EJE Z''; J
190  PRINT
200  NEXT I
```

En este bucle se calculan el módulo de cada vector y los cosenos de los ángulos que forman con los ejes.

En el caso de que el módulo del vector sea cero, se pasa al siguiente (instrucción 110).

En todos los casos se acumulan los resultados en la variable J.

```
210  LET J=0
220  IF M(1)=0 THEN 330
230  IF M(2)=0 THEN 330
240  LET J=(X(1) * X(2)+Y(1) * Y(2)+Z(1) * Z(2))/(M(1) * M(2))
250  IF J=0 THEN 290
260  IF J>=0.999999999 THEN 310
270  IF J<=-0.999999999 THEN 310
280  GOTO 340
290  PRINT ''LOS VECTORES SON PERPENDICULARES''
300  GOTO 340
310  PRINT ''LOS VECTORES SON PARALELOS''
320  GOTO 340
330  PRINT ''NO HAY ANGULO''
```

En estas instrucciones se calcula el ángulo entre vectores (240), analizando si éstos son perpendiculares (250) o paralelos (260 y 270).

```
340 PRINT ''MAS VECTORES (1=SI, 0=NO)''
350 INPUT Z
360 IF Z=1 THEN 30
370 END
```

En esta última parte se pregunta si hay más vectores para reiniciar o no el proceso de cálculo.

Ejercicios

11.20. Escribir un programa que permita calcular la suma de dos vectores, su módulo y los ángulos que dicha suma forma con los vectores y con los ejes de coordenadas.

11.21. Escribir un programa que permita calcular el producto vectorial de dos vectores, su módulo y los ángulos que dicho vector forma con los ejes de coordenadas.

12 Representación gráfica de funciones

Programa para dibujar gráficas de funciones

Para dibujar la gráfica de una función con papel y lápiz, debemos tener en cuenta dos cuestiones:

1.º ¿Dónde situar el centro?

2.º ¿Qué escala elegiremos en cada eje?

Para cada microordenador y cada función, la contestación a estas preguntas puede ser distinta, aunque en el programa que sigue se propone una solución relativamente general.

Suponemos un microordenador que tenga incorporada una pantalla con 22 filas y 32 columnas, y capaz de interpretar una instrucción semejante a PRINT AT o PLOT.

Programa: gráfica de la función $f(x)=x^2$.

```
 1  REM DIBUJA FUNCIONES
10  DEF FNF(X)=X ↑ 2
15  REM EXTREMO INFERIOR, EXTREMO SUPERIOR, NUMERO DE DIVISIONES EN EL EJE X Y SALTO
20  READ A, B, M, S
30  REM EXTREMO INFERIOR, EXTREMO SUPERIOR Y NUMERO DE DIVISIONES EN EL EJE Y
40  READ C, D, N
50  REM CENTRO
60  READ J, K
```

Estas instrucciones introducen los parámetros necesarios para dibujar la gráfica de la función definida en la instrucción 10.

```
70  LET Q=(B-A)/M
80  LET H=(D-C)/N
```

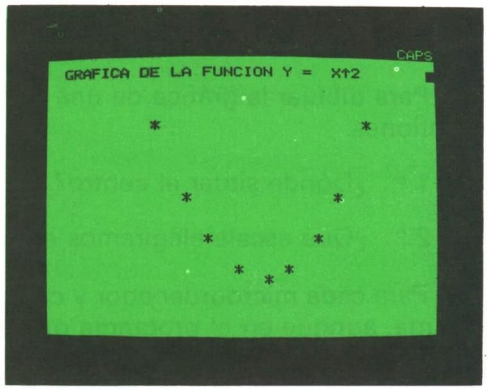
En estas instrucciones se calcula la longitud de las divisiones en cada eje.

```
 90  FOR X=A TO B STEP S
100  LET Y=FNF(X)
110  LET X1=INT ((X-A)/Q+0.5)
120  LET Y1=INT ((Y-C)/H+0.5)
130  IF J+X1 > 31 OR K-Y1 > 21 THEN 160
140  PLOT J+X1, K-Y1, " * "
150  NEXT X
155  GOTO 210
160  PRINT "MAL ESCALAS EN EJES O CENTRO"
```

Aquí se ha determinado para cada valor de X el valor de Y. A continuación se obtienen cuantas divisiones X1 e Y1 corresponden a cada valor X-A o Y-C, redondeando este valor al número entero más próximo.

En la instrucción 140 (después de verificar en la 130 que no nos hemos salido de las dimensiones adecuadas), se dibuja un asterisco (o un punto) en el lugar dado por las coordenadas J+X1 y K-Y1 (la numeración en el eje Y suele empezar de arriba abajo).

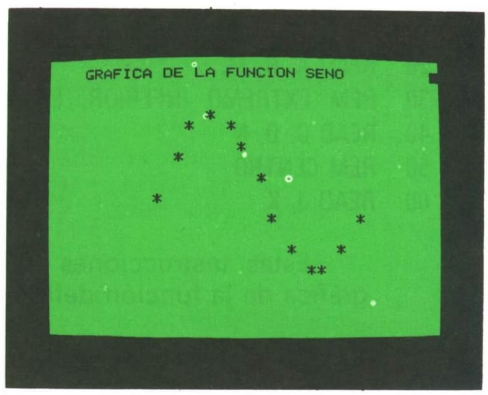
```
180  DATA -4, 4, 1, 21
190  DATA 0, 16, 15
200  DATA 10, 15
210  END
```



Estos valores introducidos en las instrucciones DATA dependen de la función que se trata de representar.

Para representar la función y=sen x, se deberían modificar simplemente las instrucciones siguientes: la 10, que quedaría así: 10 DEF FNF(X)=SIN(X), y las comprendidas entre la 180 y 200 incluidas ambas, que ahora serían:

```
180  DATA 0, 6.4, 0.5, 21
190  DATA -1, +1,15
200  DATA 10, 15
```

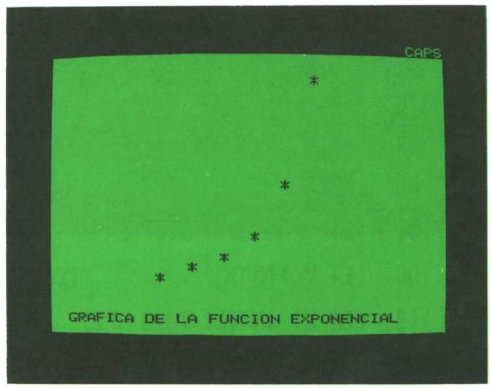


De la misma manera, para dibujar la función exponencial tendríamos que escribir:

```
10  DEF FNF(X)=EXP(X)
```

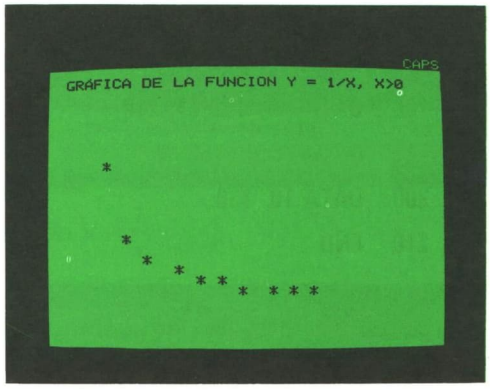
y, además,

```
180  DATA 0, 4, 0.8, 15
190  DATA 0, 28, 10
200  DATA 10, 21
```



En el caso de una hipérbola de ecuación $y=\frac{1}{x}$, las instrucciones indicadas quedarían modificadas así:

```
 10  DEF FNF(X)=1/X
180  DATA 0.3, 3, 0.3. 20
190  DATA 0.3, 4, 15
200  DATA 5, 21
```



Hay que tener en cuenta que estos valores es necesario adaptarlos a cada microordenador, de acuerdo al número de filas y columnas de que disponga su pantalla.

Gráficas en alta resolución

Las instrucciones para dibujar gráficas en alta resolución son específicas de cada microordenador, pero, a título indicativo, adaptaremos el programa anterior (diseñado para 22 filas y 32 columnas) a 100 filas y 239 columnas utilizando dos instrucciones de un microordenador determinado. (A éstas se puede encontrar sus equivalentes en otros microordenadores.)

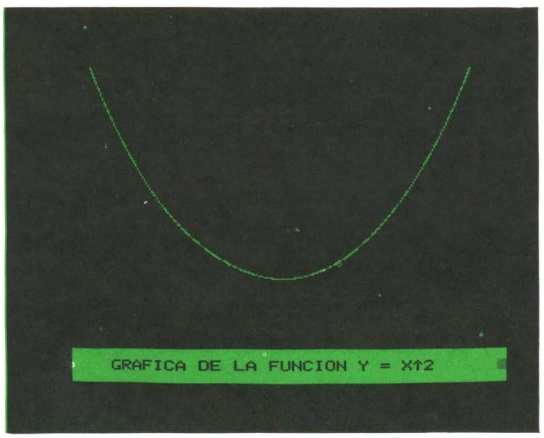
HIRES: pasa a alta resolución.
CURSET X, Y, 1: hace visible la posición X, Y, es decir, coloca un punto en la posición dada por las coordenadas X, Y.

Teniendo en cuenta estas dos instrucciones, el programa anterior que dibuja la gráfica $y=x^2$ (una parábola), quedaría así:

Programa

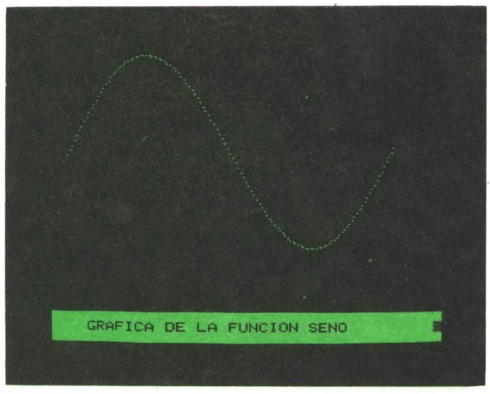
```
  1  REM ''DIBUJA FUNCIONES EN ALTA RESOLUCION''
 10  DEF FNF(X)=X ↑ 2
 20  READ A, B, S, M
 30  READ C, D, N
 60  READ J, K
 70  LET Q=(B-A)/M
 80  LET H=(D-C)/N
 85  HIRES
 90  FOR X=A TO B STEP S
100  LET Y=FNF(X)
110  LET X1=INT((X-A)/Q+0.5)
120  LET Y1=INT((Y-C)/H+0.5)
130  IF J+X1 > 239 OR K-Y1 > 100 THEN 160
140  CURSET J+X1, K-Y1, 1
150  NEXT X
155  GOTO 210
160  PRINT ''MAL ESCALAS EJES O CENTRO''
180  DATA -4, 4, 0.01, 210
190  DATA 0, 16, 150
200  DATA 10, 150
210  END
```

Se ha añadido la instrucción 85 HIRES para pasar a alta resolución, se ha modificado la instrucción 130, adaptándola al nuevo número de columnas y filas, y se ha modificado la instrucción 140, acoplándola a alta resolución. Las instrucciones DATA 180 a 200 también se han adaptado a la función $y=x^2$.



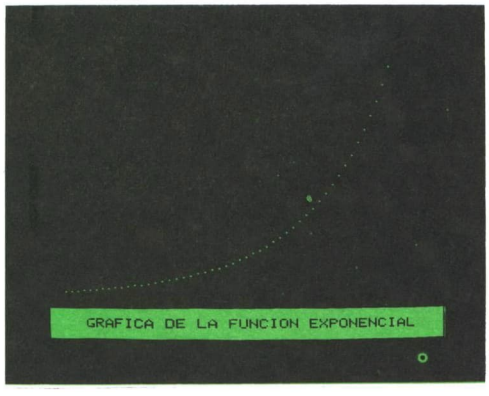
Adaptación de las instrucciones para obtener la gráfica de y=sen x

```
 10  DEF FNF(X)=SIN(X)
180  DATA 0, 6.4, 0.05, 200
190  DATA -1, +1, 150
200  DATA 10, 150
```



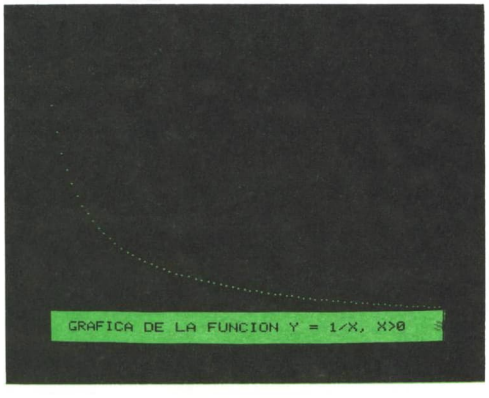
Adaptación de las instrucciones para obtener la gráfica de $y=e^x$

```
 10  DEF FNF(X)=EXP(X)
180  DATA 0, 4, 0.08, 200
190  DATA 0, 28, 100
200  DATA 10, 190
```



Adaptación de las instrucciones para obtener la gráfica de $y=\frac{1}{x}$

```
 10  DEF FNF(X)=1/X
180  DATA 0.3, 3, 0.04, 239
190  DATA 0.3, 4, 199
200  DATA 0, 199
```



Nuevamente advertimos que estos valores están adecuados a un determinado microordenador.

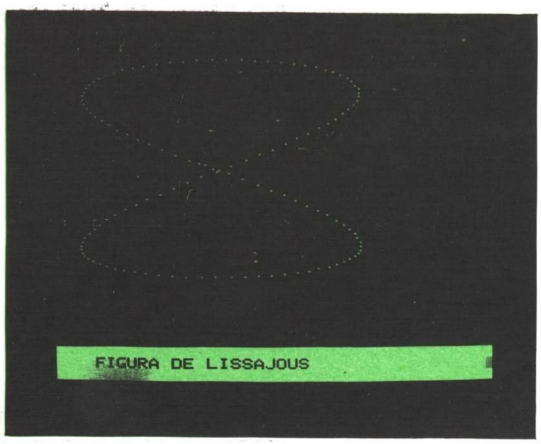
Figuras de Lissajous

Modificando el programa anterior se logra componer dos funciones sinusoidales perpendiculares, programa que puede ampliarse a cualquier composición de funciones.

Programa

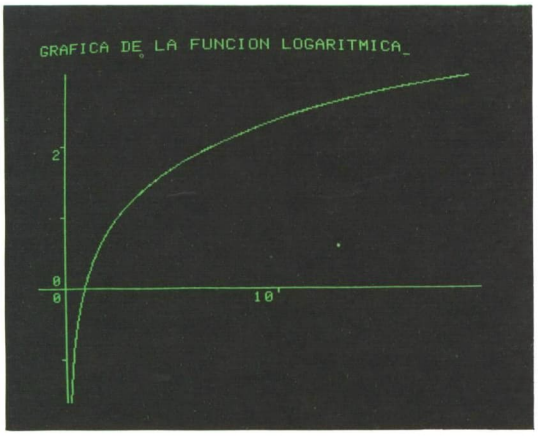
```
  2  INPUT R
  5  INTUT K
  7  INPUT M
 10  DEF FNF(X)=SIN(X)
 15  DEF FNE(X)=SIN(R * X+(2 * K+M) * PI)
 20  READ A, B, S
 30  READ C, D, N
 60  READ J, K
 80  LET H=(D-C)/N
 85  HIRES
 90  FOR X=A TO B STEP S
100  LET Y=FNF(X)
105  LET Z=FNE(X)
110  LET X1=INT ((X-C)/H+0.5)
120  LET Y1=INT ((Y-C)/H+0.5)
130  IF J+X1 > 239 OR K-Y1 > 199 THEN 160
140  CURSET J+X1, K-Y1, 1
150  NEXT X
155  GOTO 210
160  PRINT ''MAL ESCALAS O CENTRO''
180  DATA 0, 6.4, 0.05
190  DATA -1, +1, 150
200  DATA 15, 150
210  END
```

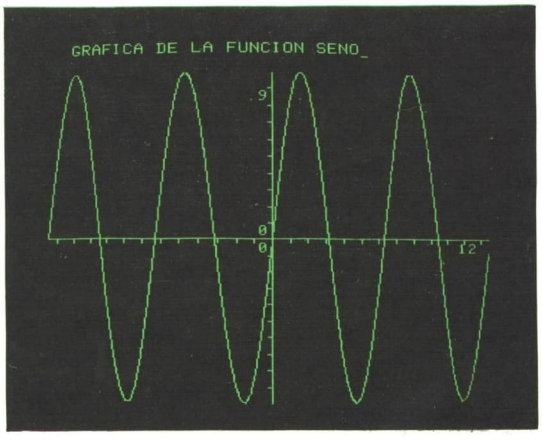
En este programa se ha definido una nueva función FNE(X) (instrucción 15), que puede modificarse variando los valores R, J y M (instrucciones 2, 5 y 7); FNE(X) se dibuja en el eje de las X (instrucción 110), utilizando la misma escala que la utilizada en el eje Y.

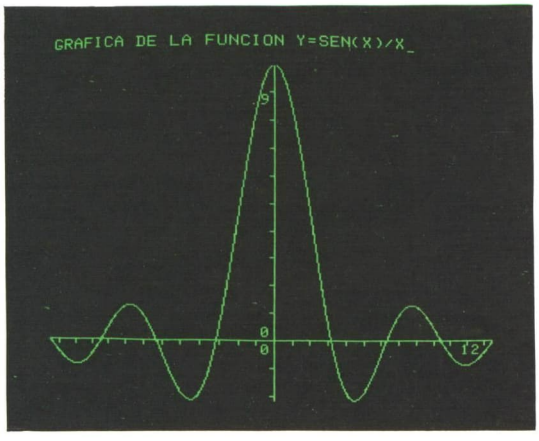


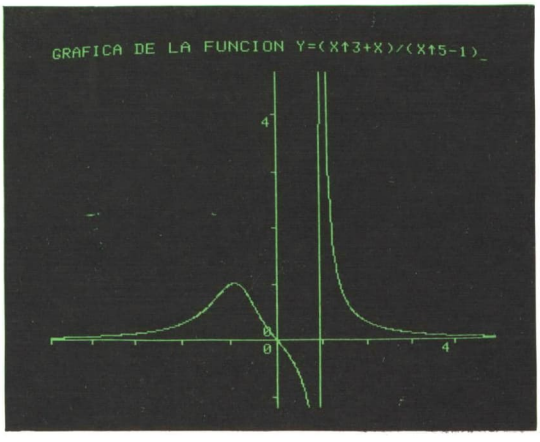
NOTA:

En las gráficas anteriores no hemos incluido los ejes coordenados para no alargar excesivamente los programas. Dichos ejes se pueden dibujar sin demasiada dificultad; así se ha hecho en los programas que dibujan las gráficas de las siguientes curvas.









Ejercicios

12.1. Escribir programas que dibujen la gráfica de las funciones:
a) y=log x
b) y=tag x
c) y=arctag x

12.2. Escribir un programa que dibuje la gráfica de la función $y=x^3$.

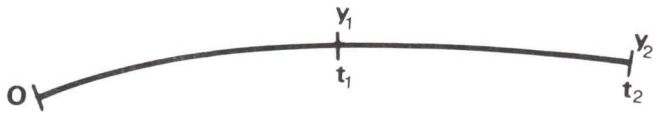
12.3. Escribir un programa que permita dibujar la gráfica de la función y=sen x+ +sen 0.9 x.

13 Ciencias

Análisis del concepto de velocidad instantánea (derivada de una función en un punto)

Consideremos un móvil que se mueve siguiendo una trayectoria cualquiera. Su velocidad media entre dos instantes t_1 y t_2, en los que ocupa dos posiciones y_1 e y_2 (con respecto a un punto de referencia O), se obtendrá dividiendo el espacio recorrido por el móvil en dicho intervalo de tiempo por el intervalo, esto es,

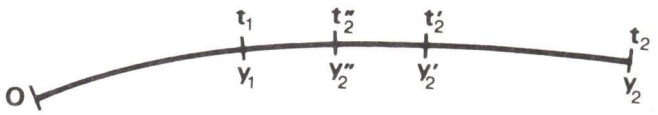
$$v_m=(y_2-y_1)\ /\ (t_2-t_1)$$



Otra velocidad que se puede considerar en este movimiento es la velocidad instantánea (la que marcaría el velocímetro de un coche), que puede obtenerse hallando el límite al que tienden la sucesión de velocidades medias cuando los instantes t_1 y t_2 tienden a confundirse en uno solo.

Dicha sucesión sería:

$$\frac{y_2-y_1}{t_2-t_1},\ \frac{y'_2-y_1}{t'_2-t_1},\ \frac{y''_2-y_1}{t''_2-t_1},\ \ldots$$



En el programa que se incluye a continuación se realiza este proceso para comprobar finalmente que los valores de las velocidades medias se aproximan a uno determinado, que es la velocidad instantánea (o derivada de la función que da el espacio con respecto al tiempo en ese instante).

El procedimiento permite calcular la derivada de cualquier función en un punto con aproximación suficiente.

Programa

```
 1  REM VELOCIDAD INSTANTANEA
10  DEF FNS (T)=T ↑ 2+2 ↑ T+3
20  INPUT T1
```

En estas instrucciones se define la función del espacio con respecto al tiempo y se introduce el instante en el que se va a obtener la velocidad.

```
30  FOR N=1 TO 15
40  T=T1+0.5 ↑ N
50  V=(FNS(T)-FNS(T1))/(T-T1)
60  PRINT INT ((T * 1000+0.5)/1000), INT ((V * 1000+0.5)/1000)
70  NEXT N
80  END
```

En esta última parte se van obteniendo los distintos términos de la sucesión de velocidades medias redondeando a tres cifras decimales. Las distintas aproximaciones de T a T1 se obtienen sumando a T1 los valores cada vez más pequeños obtenidos al elevar 0.5 a los diversos valores de N.

Ejercicios

13.1. Modificar el programa anterior para obtener las derivadas de las siguientes funciones.

a) $y=x^2+\cos^2 x$

b) $y=e^{x^2}$

c) $y=\dfrac{5x^2-3x}{2x+1}$

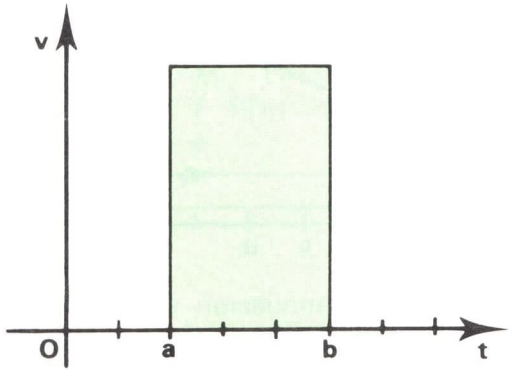
Cálculo del espacio recorrido en un movimiento (cálculo numérico de integrales definidas)

En un movimiento rectilíneo y uniforme el espacio recorrido entre dos instantes a y b viene expresado por

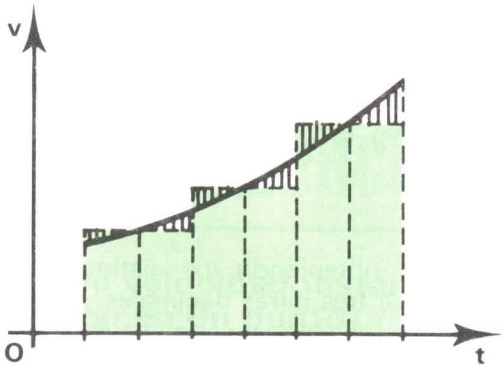
$$s=v\ (b-a)$$

donde s es el espacio recorrido, v es la velocidad (constante) y b−a, el intervalo de tiempo transcurrido

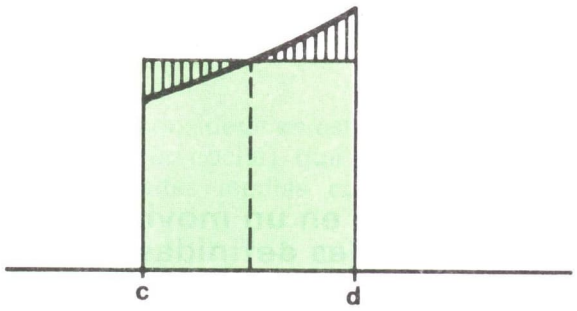
En un diagrama *velocidad-tiempo* el espacio recorrido corresponderá al área del rectángulo de altura v y base b−a.



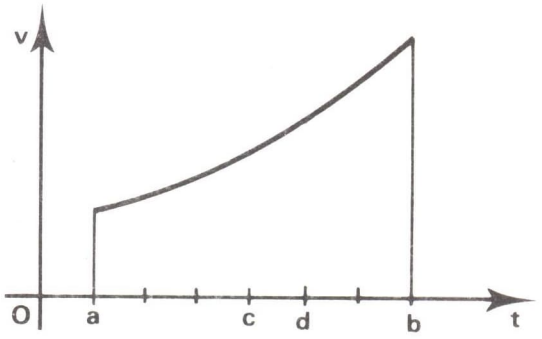
Si el movimiento no es uniforme, la gráfica en el diagrama *velocidad-tiempo* no es una línea recta paralela al eje t y el cálculo del espacio recorrido es un poco más complejo.



Dividiendo el intervalo de tiempo a–b, en intervalos parciales c–d, y aproximando el movimiento real en cada intervalo por un movimiento uniforme con velocidad constante e igual a la de dicho móvil en la mitad del intervalo (el espacio recorrido en este movimiento ficticio será el área del rectángulo rayado horizontalmente), la suma de los espacios recorridos en cada intervalo c–d, en este movimiento ficticio, se aproximará tanto más al espacio total recorrido en el movimiento real cuanto menor sea la amplitud de los intervalos c–d escogidos y más próximos estén entre sí los tramos escalonados de la figura.



Formando una sucesión de sumas, así calculadas, para intervalos de tiempo cada vez más pequeños, dicha sucesión tiende a un valor límite: el espacio recorrido por el móvil en el movimiento, o sea, la integral definida de la velocidad en el intervalo de tiempo a–b $\left(\int_b^a s(t)\ dt\right)$.



El programa que se incluye a continuación se ajusta a las ideas anteriormente expuestas y permite calcular la integral definida de cualquier función con una aproximación suficiente.

Programa

```
 1  REM ESPACIO RECORRIDO
10  DEF FNV(T)=2 * T ↑ 2+3
20  INPUT A
30  INPUT B
40  LET D=B-A
```

En esta parte del programa se define la función velocidad y se introducen los tiempos entre los cuales se va a calcular el espacio recorrido, A y B.

Se define también una variable D, que se inicia con el valor B−A, pero que se va a ir dividiendo por la mitad.

```
50  FOR I=1 TO 10
55  LET M=0
60  FOR T=A+D/2 TO B STEP D
70  LET M=M+FNV(T)
80  NEXT T
90  LET M=M * D
100 PRINT M
110 LET D=D/2
120 NEXT I
```

Se obtienen aquí 10 términos de la sucesión de sumas, que se van acumulando en la variable M. Cada suma se obtiene en el bucle interno (instrucciones 60 a 80) y en la instrucción 90.

Para cada valor de I que determina el bucle externo (50 a 120), se divide D por la mitad.

El último valor de M impreso es el más aproximado.

Ejercicios

13.2. Modificar el programa anterior para obtener las siguientes integrales definidas:

a) $\int_{-\pi}^{\pi} \operatorname{sen}^2 x\,dx$

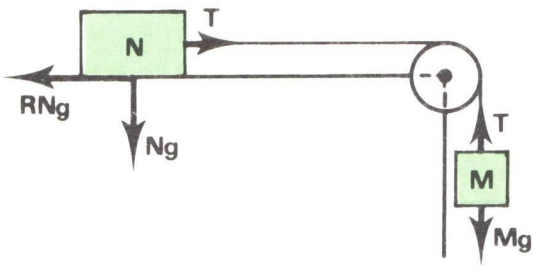
b) $\int_{0}^{2} x\cos^2 x\,dx$

c) $\int_{2}^{5} x\log x\,dx$

Cálculo de la aceleración de un sistema de dos masas

Aplicando a cada masa del sistema de la figura la segunda ley de Newton, tendremos.

$$Mg-T=MA$$
$$T-RNg=NA$$



siendo M y N los valores de las masas, T la tensión de las cuerdas, R el coeficiente de rozamiento de la masa N con la superficie horizontal y $g=9{,}8\ m/s^2$.

Resolviendo dicho sistema obtenemos como valor de la aceleración:

$$A=\frac{(M-RN)\cdot 9.8}{(M+N)}$$

Si el rozamiento es lo suficientemente grande puede ocurrir:

a) que los bloques no se muevan ($A<0$ o $A=0$ si no hay velocidad inicial), o

b) que los bloques se muevan con velocidad constante ($A=0$ si hay velocidad inicial).

El programa siguiente tiene en cuenta estas consideraciones.

Programa

```
 1  REM ACELERACION
10  PRINT "MASA 1"
20  INPUT M
30  PRINT "MASA 2"
40  INPUT N
50  PRINT "COEFICIENTE DE ROZAMIENTO"
60  INPUT R
70  PRINT "VELOCIDAD INICIAL"
80  INPUT V
```

En estas instrucciones se introducen los datos para resolver el problema.

```
90  LET A=(M-R * N) * 9.8/(M+N)
```

En la instrucción 90 se calcula el valor de la aceleración.

```
100  IF A=0 THEN 140
110  IF A < 0 THEN 160
120  PRINT "LA ACELERACION ES"; A; "M/(S ↑ 2)"
130  GOTO 170
140  IF V=0 THEN 160
150  PRINT "LA VELOCIDAD ES SIEMPRE IGUAL A"; V; "M/S"
160  PRINT "EL SISTEMA NO SE MUEVE"
170  END
```

En las instrucciones 100-110 se averigua si la aceleración es positiva; en caso afirmativo se imprime su valor (instrucción 120), y, en caso contrario, se tienen en cuenta las distintas alternativas.

Ejercicio

13.3. Ampliar el programa anterior para que calcule el espacio recorrido por los bloques en un tiempo determinado, así como su velocidad.

Análisis del movimiento de un cohete de masa variable

Consideremos un cohete cuya masa en un instante es m y que, quemando una masa Δm durante un tiempo Δt, aumenta su velocidad en Δv. Supongamos también que los gases procedentes de esta combustión tienen una velocidad u con respecto al cohete.

La fuerza media que actúa sobre el cohete $F = m\frac{\Delta v}{\Delta t}$ debe ser igual a la variación de la cantidad de movimiento del combustible expulsado.

Es decir, $m\ \Delta v/\Delta t = -u\ \Delta m/\Delta t$

Si la masa inicial del cohete es M_o y el combustible se quema a velocidad constante $\mu = \Delta m/\Delta t$, después de un tiempo t, la masa será:

$$m = M_o - \mu t$$

y sustituyendo en la ecuación anterior, se obtiene:

$$(M_o - \mu t)\ \frac{\Delta v}{\Delta t} = u\mu$$

o bien

$$\frac{\Delta v}{\Delta t} = \frac{u\mu}{M_o - \mu t}$$

Si la masa inicial del cohete es $M_o = 10000$ kg, la velocidad de combustión de los mismos es $\mu = 50\ kgs^{-1}$, y la velocidad de éstos con respecto al cohete es $u = 2.000\ ms^{-1}$, la expresión se convierte en

$$\frac{\Delta v}{\Delta t} = \frac{2000.50}{10000 - 50\ t}$$

Si el cohete está despegando de la Tierra y si suponemos que no está tan alejado de ésta para que varíe la aceleración de la gravedad, podemos escribir

$$\frac{\Delta\ v}{\Delta\ t} = \frac{100000}{10000 - 50\ t} - 9.8$$

Teniendo en cuenta que

$$\frac{\Delta v}{\Delta t}$$

se puede escribir como

$$\frac{v(t+\Delta t) - v\ (t)}{t}$$

despejando, se obtiene:

$$V\ (t+\Delta t) = v(t) + \left(\frac{100000}{10000 - 50\ t} - 9.8\right)\Delta t$$

La altura viene dada por

$$y(t+\Delta t) = y(t) + v(t)\ \Delta t$$

Si la velocidad inicial es v(0)=0 y si consideramos intervalos de tiempo de un segundo, la velocidad en la mitad del primer intervalo es

$$v(0.5)=v(0)+\left(\frac{100000}{10000-50\ t}-9.8\right)*0.5$$

Y la altura alcanzada en el primer segundo

$$y(1)=v(0.5)*1+y(0)$$

De la misma manera

$$v(1.5)=v(0.5)+\left(\frac{100000}{10000-50\ t}-9.8\right)$$

$$y(2)=v(1.5)*1+y(1)$$

y así sucesivamente

El programa siguiente permite obtener una representación gráfica de la función que da la altura alcanzada con respecto al tiempo, utilizando el procedimiento anterior y la técnica para dibujar gráficas considerada en el capítulo 12.

Programa

```
  1  REM COHETE
 10  PRINT ''TIEMPO INICIAL''
 20  INPUT T
 30  PRINT ''ALTURA INICIAL''
 40  INPUT Y
 50  PRINT ''INTERVALO DE TIEMPO''
 60  INPUT IT
 70  PRINT ''MAXIMO INTERVALO DE TIEMPO''
 80  INPUT MT
 90  PRINT ''CENTRO''
100  INPUT J
110  INPUT K
120  PRINT ''ESCALAS EN EJES''
130  INPUT L
140  INPUT N
150  LET V=0
160  HIRES
```

Hasta aquí se han introducido los datos necesarios para la ejecución del programa y en la instrucción 160 se pasa a alta resolución.

```
170  LET V=(100000/(10000-50 * IT)-9.8) * IT/2+V
```

En esta instrucción se calcula la velocidad en la mitad del primer intervalo:

```
180  U=(MT-0)/L
190  W=(100-0)/N
200  X1=INT ((T-0)/U+0.5)
210  Y1=INT ((Y-0)/W+0.5)
220  CURSET J+X1, K-Y1, 1
```

Escogiendo las escalas adecuadas se dibujan los puntos correspondientes a los valores de T e Y.

```
230  Y=Y+V * IT
240  V=V+(100000/(10000-50 * T)-9.8) * IT
250  T=T+IT
260  IF T <=MT THEN 200
270  END
```

En estas instrucciones se calculan las alturas y las velocidades para diversos tiempos hasta llegar a MT.

Datos

T=0; J=10; L=200; Y=0; K=150; N=150; IT=1; MT=20.

Ejercicios

13.4. Modificar el programa anterior de tal manera que sirva para imprimir alturas y tiempos en lugar de obtener la gráfica $y-t$.

13.5. Modificar el programa anterior de tal manera que obtenga las velocidades y represente gráficamente la función velocidad respecto al tiempo.

13.6. Elaborar un programa que dibuje la gráfica de la función altura-tiempo para el movimiento de un objeto en caída libre.

Cálculo del período de revolución de un satélite

El período T de revolución de un satélite en órbita viene expresado por la fórmula

$$T=2\pi\sqrt{\frac{(R+A)^3}{g\,R^2}}$$

siendo R el radio de la Tierra, que se supone igual a 6.370.000 metros, g=9,8 m/s^2, π=3,14159 y A la altura sobre la superficie terrestre (Para que la fórmula dé resultados correctos A debe ser mayor que 200 km y menor que 50.000 km.)

El programa que se incluye a continuación tiene en cuenta estos aspectos e imprime el período de revolución en horas y minutos.

Programa

```
 1  REM PERIODO DE REVOLUCION
10  LET G=9.8
20  LET R=63700000
30  LET PI=3.14159
40  PRINT "ALTURA DEL SATELITE EN KM"
50  INPUT A
60  IF A > 200 AND A < 50.000 THEN 90
70  PRINT "DATO INCORRECTO"
80  GOTO 50
```

En estas instrucciones se han asignado valores a las variables G, R y PI (no es necesario definir el valor de PI si el microordenador tiene esta función) y se ha introducido después el valor de la altura, comprobándose (instrucción 60) si es una altura permitida.

```
 90  LET A=A * 1000
100  LET T=2 * PI * SQR ((R+A) ↑ 3/(G * R * R))
110  LET H=INT(T/3600)
120  LET MN=INT(T-3600 * H)/60)
130  PRINT ''PERIODO DE REVOLUCION''
140  PRINT H; ''HORAS''; MN; ''MINUTOS''
```

En esta parte del programa se ha calculado el período de revolución del planeta (instrucción 100). Previamente, en la instrucción 90 se había cambiado la altura a metros. En las instrucciones 110 y 120 se ha transformado el período de revolución (obtenido inicialmente en segundos) en horas y minutos.

```
150  PRINT ''HAY OTRA ALTURA''
160  INPUT S$
170  IF S$=''SI'' THEN 40
180  END
```

Finalmente se pregunta si se introducirá una nueva altura para, en caso afirmativo, empezar de nuevo el cálculo.

Ejercicio

13.7. Escribir un programa que permita calcular la altura a la que está un satélite cuyo período de revolución se conoce.

Análisis de las ondas en una cuerda

Cuando en un medio lineal, como por ejemplo una cuerda, se establece en un punto (el foco) una vibración armónica, ésta se propaga por la cuerda y cada punto de ella vibra con un movimiento vibratorio armónico pero desfasado con respecto al foco.

La ecuación que representa el movimiento de todos los puntos de la cuerda es:

$$Y = A \operatorname{sen}\left(\frac{2\pi}{T} t - \frac{2\pi}{\lambda} R\right)$$

siendo Y la elongación de cada punto, A la amplitud, T el período (tiempo transcurrido para realizar una oscilación), λ la longitud de onda, t el tiempo transcurrido y R la distancia al foco.

En la ecuación se aprecia una doble periodicidad: con respecto al tiempo, ya que cuando t=T (período) el movimiento en un punto se repite, y con respecto a la distancia, ya que los puntos situados a 1 λ, 2λ ... vibran en fase.

A esta distancia λ se llama **longitud de onda.**

Se llama frecuencia F al número de oscilaciones que experimenta un punto del medio por unidad de tiempo, verificándose que F=1/T.

Por otro lado, existe una relación entre la longitud de onda y el período, y es la siguiente:

$$\lambda = vT$$

en la que v es la velocidad de propagación de la perturbación (que depende de las características del medio y no del movimiento del foco).

Sustituyendo en la ecuación que da Y, se obtiene:

$$Y = A \text{ sen } (2\pi Ft - 2\pi FR/v)$$

El programa que consideramos a continuación permite analizar la periodicidad frente a la distancia y, con pequeñas modificaciones, la periodicidad respecto al tiempo.

Programa

```
  1 REM ONDAS
  5 PRINT "AMPLITUD"
 10 INPUT A
 15 PRINT "FRECUENCIA"
 20 INPUT F
 25 PRINT "VELOCIDAD DE PROPAGACION"
 30 INPUT V
 35 PRINT "LONGITUD DE LA CUERDA"
 40 INPUT MR
 45 PRINT "INTERVALO DE LONGITUD"
 50 INPUT D
 55 PRINT "CENTRO"
 60 INPUT J
 65 INPUT K
 70 PRINT "ESCALAS"
 75 INPUT L
 80 INPUT N
 90 PRINT "TIEMPO"
100 INPUT T
```

En este conjunto de instrucciones se introducen los datos necesarios para que el microordenador dibuje una gráfica de la elongación Y en función de la distancia.

```
105 HIRES
110 LET U=(MR-0)/L
120 LET W=(A-(-A))/N
```

En estas instrucciones se determina la longitud de las escalas sobre cada eje.

```
130 FOR R=0 TO MR STEP D
140 LET Y=A * SIN (2 * PI * F * T-2 * PI * F * R/V)
150 LET X1=INT ((R-0)/U+0.5)
160 LET Y1=INT ((Y+A)/W+0.5)
170 CURGET J+X1, K-Y1, 1
180 NEXT R
190 END
```

En este bucle, previo cálculo del valor de la elongación para el instante considerado T y en cada punto de la cuerda, se determinan los puntos que se van a representar (instrucciones 150 y 160) y se dibuja cada punto haciendo visible la posición del cursor.

Datos

A=0.3 m, F=800 Hz, V=1200 ms^{-1}, MR=1.5 m.,
D=0.15, J=10, K=150, L=200, N=150, T= ————

Ejercicios

13.8. Modificar el programa anterior para que se impriman los valores de R e Y redondeados a tres cifras decimales.

13.9. Modificar el programa anterior de tal manera que permita observar la periodicidad de Y frente a T para un punto cualquiera de la cuerda.

Estudio de las interferencias

Un punto de un medio, por ejemplo de una cuerda, alcanzado simultáneamente por dos movimientos ondulatorios, vibra con un movimiento que es suma del que producirían cada uno de ellos independientemente.

Consideremos el caso de dos movimientos ondulatorios de la misma amplitud y frecuencia. La interferencia se produce debido a la diferencia de caminos recorridos.

La ecuación que representa el movimiento de cada punto de la cuerda puede obtenerse sumando la función de onda de cada uno de los movimientos:

$$Y=Y_1+Y_2=A \text{ sen } (2\pi FT-2\pi FR1/V)+A \text{ sen } (2\pi FT-2\pi FR2/V)$$

El programa siguiente demuestra que si $R2-R1=K\lambda$ (K=1, 2, 3...), el efecto de la interferencia corresponde a un máximo, y si $R2-R1=(2K-1)\lambda/2$, (K=1, 2, 3...) se produce una interferencia destructiva.

Programa

```
  1 REM INTERFERENCIAS
  5 PRINT "AMPLITUD"
 10 INPUT A
 15 PRINT "FRECUENCIA"
 20 INPUT F
 25 PRINT "VELOCIDAD DE PROPAGACION"
 30 INPUT V
 35 PRINT "TIEMPO INICIAL"
 40 INPUT T
 45 PRINT "INTERVALO DE TIEMPO"
 50 INPUT IT
 55 PRINT "MAXIMO INTERVALO DE TIEMPO"
 60 INPUT MT
 65 PRINT "CENTRO"
 70 INPUT J
 75 INPUT K
 80 PRINT "ESCALAS"
 85 INPUT L
 90 INPUT N
 95 PRINT "DISTANCIAS AL FOCO"
100 INPUT R1
110 INPUT R2
115 HIRES
```

En estas instrucciones se introducen los datos necesarios y se obliga al microordenador a imprimir en alta resolución.

```
120  LET Y=A * SIN (2 * PI * F * T-2 * PI * F * R1/V)+A * SIN (2 * PI * F * T-2 * PI * F * R2/V)
130  LET U=(MT-0)/L
140  LET W=(2 * A-(-2 * A))/N
150  LET X1=INT ((T-0)/U+0.5)
160  LET Y1=INT ((Y-(-2 * A))/W+0.5)
165  CURSET J+X1, K-Y1, 1
170  LET T=T+IT
180  IF T<=MT THEN 120
190  END
```

Se calculan los diferentes valores de Y (instrucción 120), la separación entre divisiones en cada eje (instrucciones 130, 140) y los valores a representar en cada eje (instrucciones 150-160). En la instrucción 170 se obtiene un nuevo valor de T y en la 180 se decide la continuación del proceso.

Datos

A=0.3 m, F=50 Hz, V=340 ms^{-1}, T=0, IT=0.001, MT=0.06, J=10, K=150, L=200, N=150, R1= ———, R2= ——— (λ=6.8 m)

Ejercicio

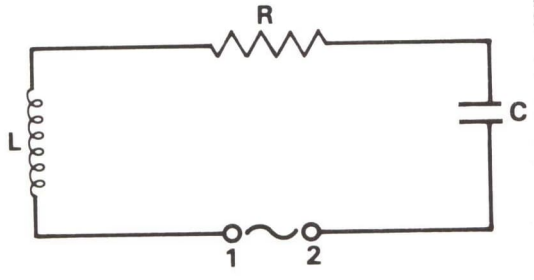
13.10. Modificar el programa anterior para que imprima tiempos y elongaciones.

Estudio de las oscilaciones en un circuito eléctrico

Consideremos un circuito con un condensador C, una resistencia R y una autoinducción L, dispuestos en serie. El voltaje V(t) aplicado en los extremos 1 y 2 es el trabajo necesario para trasladar la unidad de carga desde el punto 1 al 2 y debe ser igual a la suma de:

— la caída de potencial experimentada en los extremos de la bobina

$$V_L = L\frac{dI}{dt} = L\frac{d^2Q}{dt^2}$$



— la caída de potencial experimentada en los extremos de la resistencia

$$V_R = RI = R\,dQ/dt$$

— la caída de potencial experimentada en los extremos del condensador

$$V_c = Q/C$$

Así, podremos escribir:

$$L\, d^2Q/dt^2 + R\, dQ/dt + Q/C = V(t)$$

Si el voltaje varía sinusoidalmente con el tiempo $V(t) = Vo \text{ sen } (2 \pi Ft)$ (donde Vo es el voltaje máximo, F es la frecuencia y t el tiempo transcurrido), la ecuación anterior queda modificada así:

$$L\frac{dI}{dt} + RI + \frac{Q}{C} = Vo \text{ sen } (2 \pi FT)$$

Despejando, se obtiene

$$\frac{dI}{dt} = Vo/L \text{ sen } (2 \pi Ft) - RI/L - Q/LC$$

Y en forma de incrementos finitos se tendrá:

$$\Delta I/\Delta t = Vo/L \text{ sen } (2 \pi Ft) - RI/L - Q/LC$$

O bien, teniendo en cuenta que

$$\frac{\Delta I}{\Delta t} = \frac{I(t+t) - I(t)}{t}$$

se obtiene:

$$I(t+\Delta t) = I(t) + (Vo/L \text{ sen } (2 \pi Ft) - RI/L - Q/LC)\, \Delta t$$

Pero como la intensidad $I = \Delta Q/\Delta t$, se llega finalmente a $Q(t+\Delta t) = Q(t) + I \cdot \Delta t$.

El programa siguiente tiene en cuenta las expresiones anteriores; calcula la carga para intervalos de tiempo sucesivos y la intensidad en la mitad de dichos intervalos. Así, por ejemplo, para intervalos 0–0.1, 0.1–0.2, se obtiene:

$$I(0.05) = I(0) + Vo/L * \text{sen } (2\pi FT \times 0.1) - (RI(0)/L) \cdot (0.1) - Q(0)/LC) \times (0.1)$$

y

$$Q(0.1) = Q(0) + I(0.05) \times (0.1)$$

De la misma manera

$$I(0.15) = I(0.05) + Vo/L \text{ sen } (2\pi Ft \times 0.1) - (RI(0.05)/L) \times 0.1 - Q(0.05)/LC) \times 0.1$$

y

$$Q(0.2) = I\ (0.15) \times (0.1) + Q\ (0.1)$$

Dicho programa permite obtener una representación gráfica de la intensidad de corriente frente al tiempo utilizando las técnicas expuestas en el capítulo 12, relativo a gráficas.

Programa

```
  1 REM OSCILACIONES
  5 PRINT ''TIEMPO INICIAL''
 10 INPUT T
 15 PRINT ''INTENSIDAD INICIAL''
 20 INPUT I
 25 PRINT ''CARGA INICIAL''
 30 INPUT Q
 35 PRINT ''FRECUENCIA''
 40 INPUT F
 45 PRINT ''VOLTAJE EFICAZ''
 50 INPUT V
 55 PRINT ''RESISTENCIA, INDUCTANCIA, CAPACIDAD''
 60 INPUT R
 70 INPUT L
 80 INPUT C
 90 PRINT ''INTERVALO DE TIEMPO''
 95 INPUT IT
100 PRINT ''MAXIMO INTERVALO DE TIEMPO''
105 INPUT MT
```

Hasta aquí se introducen los datos necesarios para los cálculos (hay que tener en cuenta que el voltaje eficaz es $V_{ef}=V_o \cdot \sqrt{2}$).

```
110 PRINT ''MAXIMA INTENSIDAD''
120 INPUT I0
130 PRINT ''CENTRO''
135 INPUT J
140 INPUT K
145 PRINT ''ESCALAS''
150 INPUT H
155 INPUT N
160 HIRES
```

Se introducen los datos necesarios para situar la gráfica de la función solución en la pantalla y se pasa al modo de alta resolución.

```
170 LET I=V * SQR(2) * SIN(2 * PI * F * T) * IT/(2 * L)-R * I * IT/(2 * L)-Q * IT/(2 * L * C)+I
```

En esta instrucción se calcula el valor de la intensidad en la mitad del primer intervalo de tiempo.

```
180 LET U=(MT-0)/H
190 LET W=(I0-(-I0))/N
200 LET X1=INT((T-0)/U+0.5)
210 LET Y1=INT((I-(-I0))/W+0.5)
220 CURSET J+X1,K-Y1,1
```

Se calcula la longitud de las divisiones de los ejes, se ajusta la escala y se dibujan las intensidades frente a los tiempos, haciendo visible el punto del cursor.

```
230  Q=Q * IT+Q
240  I=V * SQR(2) * SIN(2 * PI * F * T) * IT/L-R * I * IT/L-Q * IT/(L * C)+I
250  T=T+IT
260  IF T <=MT THEN 200
270  END
```

En la instrucción 230 se calcula la carga en los extremos de cada intervalo de tiempo, mientras en la 240 se calcula la intensidad en el punto medio de cada intervalo.

Las instrucciones 250 y 260 permiten la continuación del proceso hasta el valor MT.

Datos

T=0, I=0, Q=0, F=50, V=220, R=20, L=3E-2, C=2E-3, IT=0.001, MT=0.06, IO=15, J=10, K=150, H=200, N=150, TO=15.

Ejercicios

13.11. Modificar el programa anterior para que imprima resultados en lugar de dibujar la gráfica.

13.12. Ejecutar el programa anterior utilizando la frecuencia de resonancia $F=\frac{1}{2}\sqrt{LC}$ y comparar los resultados obtenidos para frecuencias mayores y menores.

13.13. Obtener un programa que permita el estudio de las oscilaciones eléctricas amortiguadas (sin el voltaje aplicado V * SIN(2 * PI * F * T).

Simulación de la desintegración radiactiva

Supongamos una determinada cantidad de monedas que lanzamos al aire. Una vez que éstas han caído, quitamos las que presentan «cruz» anotando las «caras». Lanzamos al aire las monedas restantes (las «caras») retirando las «cruces» y anotando de nuevo las que han salido «cara». Repetimos el procedimiento varias veces.

La variación en cada lanzamiento del número de caras puede utilizarse como modelo para analizar la variación en el tiempo del número de átomos que no se desintegran en un proceso radiactivo.

El programa que se incluye a continuación permite hacer esta simulación con la ayuda de un microordenador.

Programa

```
 10  PRINT "NUMERO DE MONEDAS"
 20  INPUT N
 30  LET C=0
 40  LET D=0
 50  FOR I=1 TO N
 60  LET S=INT(RND(1)+0.5)
 70  IF S=1 THEN 100
 80  LET D=D+1
 90  GOTO 110
100  LET C=C+1
110  NEXT I
120  PRINT C
```

Introducido el número de monedas N, se definen dos variables C y D que servirán para contar el número de caras (C) y cruces (D) que existen en cada momento.

El bucle 50 a 110 simula el lanzamiento de N monedas, lanzando N veces una moneda.

```
130  LET N=N-D
140  IF N<=2 THEN 160
150  GOTO 30
160  END
```

La instrucción 130 resta al número de monedas N el número de cruces D, antes de iniciar un nuevo lanzamiento. El proceso se detiene cuando el número de monedas es menor o igual que 2 (instrucción 140).

Utilización del sistema periódico de los elementos

Pretendemos en este programa:

A) Introducir la tabla periódica de los elementos, en la que se incluyen:

- — Símbolo del elemento (dos caracteres).
- — Número atómico (tres caracteres).
- — Peso atómico (seis caracteres).
- — Configuración electrónica, en la que se incluye el símbolo del gas noble más próximo, entre paréntesis, y lo que le diferencia de él. Así, por ejemplo, la configuración electrónica del radón es (Xe)4f14 5d10 6s2 6p6 (hasta un total de veinte caracteres).

B) Conseguir que, dado el símbolo de un elemento, aparezcan en pantalla:

1.º Todas sus características.

2.º Número atómico.

3.º Peso atómico.

4.º Configuración electrónica.

5.º Los elementos que se encuentran en el mismo período y grupo del elemento en cuestión.

Para construir el programa se define la variable E$(I) concatenando: la variable A$(I) que contiene el símbolo del elemento (dos caracteres como máximo), B$(I) que contiene el número atómico (hasta tres caracteres), C$(I) que contiene la masa atómica (hasta seis caracteres incluyendo el punto decimal), D$(I) que contiene la configuración electrónica (con veinte caracteres).

Además, se introduce una variable N(X, Y) de dimensiones (7, 18) que incluye la posición que ocupa cada elemento en el sistema periódico; tomando diversos fragmentos de la variable E$(I) se podrán localizar las diferentes características de un elemento determinado.

Para averiguar el período y el grupo de un elemento se localiza primero su número atómico y después se busca en la tabla N(X, Y).

Programa

```
  1  REM ELEMENTOS
 10  DIM E$(104)
 20  DIM A$(104)
 30  DIM B$(104)
 40  DIM C$(104)
 50  DIM D$(104)
 60  DIM A(104)
 70  DIM B(104)
 80  DIM C(104)
 90  DIM D(104)
100  DIM N(7, 18)
```

En estas instrucciones se definen las variables que se van a utilizar en el programa.

```
110  FOR I=1 TO 104
120  READ A$(I)
130  READ B$(I)
140  READ C$(I)
150  READ D$(I)
160  LET A(I)=LEN (A$(I))
170  LET B(I)=LEN (B$(I))
180  LET C(I)=LEN (C$(I))
190  LET D(I)=LEN (D$(I))
200  FOR J=A(I)+1 TO 2
210  LET A$(I)=A$(I)+ ''  ''
220  NEXT J
230  FOR K=B(I)+1 TO 3
240  LET B$(I)=B$(I)+ ''  ''
250  NEXT K
260  FOR S=C(I)+1 TO 6
270  LET C$(I)=C$(I)+ ''  ''
280  NEXT S
290  FOR T=D(I)+1 TO 20
300  LET D$(I)=D$(I)+ ''  ''
310  NEXT T
```

En este conjunto de instrucciones se completan con espacios en blanco las variables leídas para que tengan el número de caracteres fijado con anterioridad.

```
320  LET E$(I)=A$(I)+B$(I)+C$(I)+D$(I)
330  NEXT I
```

Se construye la variable E$(I) concatenando las anteriores.

```
340 FOR Y=1 TO 18
350 FOR X=1 TO 7
360 READ N(X, Y)
370 NEXT X
380 NEXT Y
390 DATA
400 DATA
```

Aquí se leen los números atómicos de cada elemento de la tabla periódica por columnas, colocando ceros en los lugares vacíos.

En las primeras instrucciones DATA se incluyen las características de cada elemento: símbolo y número atómico, peso atómico y configuración electrónica, separados por comas. Estas características se introducen por orden consecutivo de números atómicos.

En las últimas instrucciones DATA se colocan los números atómicos de cada elemento, ordenados por columnas (grupos) correspondientes a cada fila (período).

```
600 PRINT ''SI QUIERE TODAS LAS CARACTERISTICAS DE UN ELEMENTO ESCRIBA 1''
610 PRINT ''SI QUIERE EL NUMERO ATOMICO ESCRIBA 2''
620 PRINT ''SI QUIERE PESO ATOMICO ESCRIBA 3''
630 PRINT ''SI QUIERE CONFIGURACION ELECTRONICA ESCRIBA 4''
640 PRINT ''SI QUIERE GRUPO Y PERIODO ESCRIBA 5''
650 INPUT U
660 IF U=1 THEN 750
670 IF U=2 THEN 820
680 IF U=3 THEN 890
690 IF U=4 THEN 960
700 IF U=5 THEN 1030
710 PRINT ''EL PROGRAMA NO HACE ESO''
720 STOP
```

En estas instrucciones se elige la tarea que se va a realizar.

```
750 PRINT ''INTRODUZCA EL SIMBOLO DEL ELEMENTO''
760 INPUT H$
761 LET L=LEN (H$)
763 IF L=2 THEN 770
765 LET H$=H$+ ''  ''
770 FOR I=1 TO 104
780 IF H$=LEFT$ (E$(I), 2) THEN 800
790 GOTO 810
800 PRINT E$(I)
810 NEXT I
815 GOTO 600
```

Se introduce un símbolo de un elemento y, en el caso de tener un carácter, se añade un espacio; se busca después en la memoria el símbolo del elemento, separando los dos caracteres de la izquierda de E$ (I) (instrucción 780); una vez encontrado, se imprime el contenido de la variable E$ (I).

```
820  PRINT ''INTRODUZCA EL SIMBOLO DE UN ELEMENTO''
830  INPUT H$
831  LET L=LEN (H$)
833  IF L=2 THEN 840
835  LET H$=H$+ ''  ''
840  FOR I=1 TO 104
850  IF H$=LEFT$ (E$(I), 2) THEN 870
860  GOTO 880
870  PRINT MID$(E$(I), 3, 3)
880  NEXT I
885  GOTO 600
```

Aquí se imprimen los siguientes tres caracteres de E$(I) (número atómico) una vez localizado el símbolo del elemento.

```
890  PRINT ''INTRODUZCA EL SIMBOLO DE UN ELEMENTO''
900  INPUT H$
901  LET L=LEN (H$)
903  IF L=2 THEN 910
905  LET H$=H$+ ''  ''
910  FOR I=1 TO 104
920  IF H$=LEFT$(E$(I), 2) THEN 940
930  GOTO 950
940  PRINT MID$(E$ (I), 6, 6)
950  NEXT I
955  GOTO 600
```

En estas instrucciones se imprimen los seis caracteres siguientes de E$(I) (peso atómico) una vez localizado el símbolo del elemento.

```
960  PRINT ''INTRODUZCA EL SIMBOLO DE UN ELEMENTO''
970  INPUT H$
971  LET L=LEN (H$)
973  IF L=2 THEN 980
975  LET H$=H$+ ''  ''
980  FOR I=1 TO 104
990  IF H$=LEFT$(E$(I), 2) THEN 1010
1000 GOTO 920
1010 PRINT RIGHT$(E$(I), 20)
1020 NEXT I
1025 GOTO 600
```

En esta parte, una vez localizado el símbolo del elemento, se imprimen los veinte caracteres siguientes (los primeros veinte a partir de la derecha) que corresponden a la configuración electrónica.

```
1030  PRINT ''INTRODUZCA EL SIMBOLO DEL ELEMENTO''
1040  INPUT H$
1041  LET L=LEN (H$)
1042  IF L=2 THEN 1045
1043  LET H$=H$+ ''  ''
1045  LET H=0
1050  FOR I=1 TO 104
1060  LET H=H+1
1070  IF LEFT$ (E$ (I), 2)=H$ THEN 1090
1080  NEXT I
1090  FOR X=1 TO 7
1100  FOR Y= 1 TO 18
1110  IF N(X, Y)=H THEN 1150
1120  NEXT Y
1130  NEXT X
1140  PRINT ''ES UNA TIERRA RARA''
1150  FOR P=1 TO 7
1160  PRINT LEFT$ (E$ (N(P, Y)), 2)
1170  NEXT P
1180  PRINT
1190  FOR Q=1 TO 18
1200  PRINT LEFT$ (E$(X, Q), 2)
1210  NEXT Q
1215  GOTO 600
1220  END
```

Se introduce el símbolo de un elemento y se busca, considerando los dos caracteres de la izquierda de E$(I), el valor del número atómico correspondiente, que se almacena en H.

Se localiza a continuación en la tabla N(X, Y), imprimiendo los elementos del grupo y período a los que pertenece el elemento en cuestión.

Ejercicio

13.14. Modificar el programa anterior para que se incluyan otros datos como radios atómicos, potenciales de ionización, etcétera.

Igualación de ecuaciones químicas

El método de igualación de reacciones químicas que se explica a continuación es muy laborioso, pero resulta muy sencillo y rápido si se utiliza un microordenador y el programa para resolver sistemas de ecuaciones del capítulo 11, donde se desarrolla un método de

resolución de sistemas de ecuaciones lineales. Supongamos que se trata de igualar la reacción química:

$$X1\ Fe\ S_2 + X2\ O_2 \longrightarrow X3\ Fe_2\ O_3 + X4\ SO_2$$

Para que se cumpla el principio de conservación de la materia debe verificarse:

$$\begin{cases} X1 = 2X3 \\ 2X1 = X4 \\ 2X2 = 3X3 + 2X4 \end{cases}$$

Resulta así un sistema de 3 ecuaciones con 4 incógnitas (X1, X2, X3, X4). Este sistema de ecuaciones, en principio, es indeterminado; si se desea una solución, se da un valor arbitrario a una de las incógnitas, por ejemplo, X1 = 1.

Resulta así el sistema cuadrado (4 ecuaciones con 4 incógnitas), el cual se puede resolver por el método explicado en el capítulo 11.

$$\begin{cases} X1 - 2X3 = 0 \\ 2X1 - X4 = 0 \\ 2X2 - 3X3 - 2X4 = 0 \\ X1 = 1 \end{cases}$$

Resuelto con la ayuda del microordenador, da los siguientes valores:

$$X1 = 1;\ X2 = 2{,}75;\ X3 = 0{,}5;\ X4 = 2$$

Multiplicamos estos valores por 4 para obtener coeficientes enteros:

$$2\ Fe\ S_2 + 11\ O_2 \longrightarrow 2\ Fe_2\ O_3 + 8\ SO_2$$

De la misma manera, en la reacción

$$X1\ K_2\ Cr_2\ O_7 + X2\ H\ Cl \longrightarrow X3\ K\ Cl + X4\ Cr\ Cl_3 + X5\ H_2O + X6\ Cl_2$$

deben verificarse estas ecuaciones:

$$\begin{cases} 2X1 = X3 \\ 2X1 = X4 \\ 7X1 = X5 \\ X2 = 2X5 \\ X2 = X3 + 3X4 + 2X6 \end{cases}$$

Resulta un sistema de 5 ecuaciones con 6 incógnitas, que se puede reducir a un sistema cuadrado de 6×6, dando a una incógnita cualquiera un valor arbitrario; por ejemplo, asignamos a X1 el valor 1:

$$\begin{cases} 2X1 - X3 = 0 \\ 2X1 - X4 = 0 \\ 7X1 - X5 = 0 \\ X2 - 2X5 = 0 \\ X2 - X3 - 3X4 - 2X6 = 0 \\ X1 = 1 \end{cases}$$

Resolviendo este sistema con la ayuda del microordenador, da los resultados siguientes:

$$X1=1;\ X2=14;\ X3=2;\ X4=2;\ X5=7;\ X6=3$$

lo que permite escribir:

$$K_2Cr_2O_7 + 14\ HCl \longrightarrow 2\ KCl + 2CrCl_3 + 7H_2O + 3Cl_2$$

Ejercicio

13.15. Igualar, con la ayuda del microordenador, la reacción química

$$MnO_2 + HCl \longrightarrow MnCl_2 + H_2O + Cl_2$$

Valoración ácido fuerte. Base fuerte

La ecuación (no simplificada) que expresa el equilibrio *ácido fuerte-base fuerte* viene dada por la expresión

$$x^2+(C_B-C_A)\ x-K_E=0$$

siendo x la concentración molar de iones H_3O^+ presentes en la disolución, C_B y C_A las concentraciones molares de la base y el ácido en cada etapa de disolución, y K_E el producto iónico del agua, que depende de la temperatura.

El propósito de este programa es estudiar la variación del pH en el curso de una valoración de un ácido fuerte, por ejemplo, ácido clorhídrico con una base fuerte, como puede ser hidróxido de sodio, o a la inversa.

De las dos raíces de la ecuación sólo nos interesa la positiva. Sin embargo, debido al pequeño valor de K_E (10^{-14} a 25°C) y al número de cifras significativas que maneja un microordenador, podría encontrarse que la raíz cuadrada llegara a tomar el valor cero dando una solución de x negativa; ésta, al calcular el logaritmo ($pH=-\log(H_3O^+)$), produciría error, dado que no existen logaritmos decimales de números negativos.

Para solucionar este problema, calculamos la mayor de las dos raíces en valor absoluto. Si es positiva, es la que buscamos, y si es negativa, calculamos el producto de las raíces ($x_1 \cdot x_2=c/a=K_E$) para obtener la otra.

Programa

```
10 PRINT "CARACTERISTICAS DE LA DISOLUCION A VALORAR"
20 PRINT "SI ES ACIDA ESCRIBIR 1, SI ES BASICA ESCRIBIR 2"
30 INPUT P
40 IF P=1 OR P=2 THEN 60
50 GOTO 30
```

En este grupo de instrucciones se determinan las características de la disolución a valorar.

```
 60  PRINT ''MOLARIDAD DEL PRODUCTO INICIAL''
 70  INPUT M1
 80  IF M1 < 0.001 OR M1 > 1 THEN 100
 90  GOTO 120
100  PRINT ''DATOS INCORRECTOS''
110  GOTO 70
120  PRINT ''VOLUMEN INICIAL EN MILILITROS''
130  INPUT V1
140  IF VI < 0 OR V1 > 100 THEN 160
150  GOTO 180
160  PRINT ''DATO INCORRECTO''
170  GOTO 130
180  PRINT ''DISOLUCION UTILIZADA EN LA VALORACION''
190  PRINT ''MOLARIDAD''
200  INPUT M2
210  IF M2 < 0.001 OR M2 > 1 THEN 230
220  GOTO 250
230  PRINT ''DATO INCORRECTO''
240  GOTO 200
250  PRINT ''VOLUMEN TOTAL VERTIDO EN MILILITROS''
260  INPUT VT
270  IF VT < =1 OR VT > 10 * V1 THEN 290
280  GOTO 310
290  PRINT ''DATO INCORRECTO''
300  GOTO 260
310  PRINT ''CUANTOS MILILITROS SE AÑADEN CADA VEZ''
320  INPUT IV
330  IF IV < VT/20 OR IV > VT/5 THEN 350
340  GOTO 370
350  PRINT ''DATO INCORRECTO''
360  GOTO 320
```

Hasta aquí se han introducido los datos necesarios verificando su corrección.

En las instrucciones 80, 140, 270, 330 se han establecido condiciones habituales en un laboratorio.

```
370  PRINT
380  PRINT ''V'', ''PH''
390  FOR V=0 TO VT STEP IV
400  LET C1=M1 * V1/(V1+V)
410  LET C2=M2 * V/(V1+V)
420  IF P=1 THEN 440
430  GOTO 470
440  LET CA=C1
450  LET CB=C2
460  GOTO 490
470  LET CA=C2
480  LET CB=C1
490  GOSUB 600
500  PRINT V, PH
510  NEXT V
520  STOP
```

En estas instrucciones se calculan las concentraciones de las sustancias, teniendo en cuenta que sean ácido o base, y se envía el control del programa, cada vez que es necesario, a la subrutina 600 para calcular el pH.

```
600  LET KE=1E-14
610  LET D=SQR ((CB-CA) ↑ 2+4 * KE)
620  LET H1=(-(CB-CA)+D)/2
630  LET H2=(-(CB-CA)-D)/2
640  IF (CB-CA) < 0 THEN 670
650  LET H=-KE/H2
660  GOTO 680
670  LET H=H1
680  LET PH=-LOG(H)/2.3
690  RETURN
700  END
```

En esta subrutina se calculan las soluciones de la ecuación, teniendo en cuenta los comentarios que se hicieron en la introducción del programa.

Posteriormente se calcula el pH ($pH = -\log [H_3O^-] = -LOG\ [H]/2.3$.

En la instrucción 680 LET PH=-LOG(H)/2.3, la función LOG ( ) calcula el logaritmo neperiano, de ahí que haya que dividir por la constante 2.3 para obtener el logaritmo decimal (*).

Ejercicio

13.16. Realizar un programa que permita analizar el equilibrio químico entre productos gaseosos.

Cálculo del número de células nacidas en un determinado tiempo

Las células que se reproducen dividiéndose en dos pueden, si el alimento es adecuado, duplicar su número en intervalos regulares de tiempo.

La relación $\frac{dN}{dt} = N$, que señala que la variación del número de células por unidad de tiempo es igual al número de células existentes en un instante, representa bien este fenómeno, ya que si se escribe en forma de incrementos finitos se obtiene:

$$\frac{\Delta N}{\Delta t} = N \qquad \text{o bien} \qquad \frac{N(t+\Delta t)-N(t)}{\Delta t} = N$$

Y despejando se llega a

$$N(t+\Delta t) = N(t) \times \Delta t + N(t)$$

(*) La fórmula de transformación de logaritmos naturales a decimales es:

$$\log x = \ln x \log e = \ln x \frac{1}{\frac{1}{\log e}} = \frac{\log x}{2.3}, \text{ siendo } 1/\log e = 2.3$$

Si, por ejemplo, $N(0)=1$ y $\Delta t=1$

$$N(1)=N(0)\times 1+N(0)=1+1=2$$
$$N(2)=N(1)\times 1+N(1)=2+2=4$$
$$N(3)=N(2)\times 1=N(2)=4+4=8$$
$$\ldots\ldots$$
$$\ldots\ldots$$

Es decir, el número de células se duplica en cada unidad de tiempo.

Programa

```
 1  REM CELULAS
10  PRINT ''INTRODUZCA EL NUMERO INICIAL DE CELULAS''
20  INPUT N
30  PRINT ''TIEMPO INICIAL''
40  INPUT T
50  PRINT ''INTERVALO DE TIEMPO''
60  INPUT IT
70  PRINT ''MAXIMO INTERVALO DE TIEMPO''
80  INPUT MT
```

En estas instrucciones se introducen los datos necesarios para resolver el problema.

```
 90  PRINT T, N
100  LET N=N * IT+N
110  LET T=T+IT
120  IF T<=MT THEN 90
130  END
```

— En la instrucción 90 se imprimen el tiempo y el número de células que existen en ese instante.
— En la instrucción 100 se determina el número de células existentes en un instante a partir de las existentes en el instante anterior.
— En la 110 se incrementa en IT el tiempo. Si éste es menor o igual que MT se repite el proceso (120), y si no es así, se detiene.

Datos

N=1; T=0; IT=1; MT=30

El programa anterior es válido, con muy ligeras modificaciones, para procesos como la desintegración radiactiva, la descarga de un condensador, es decir, aquéllos en que dos magnitudes F y x están relacionadas por una ecuación del tipo

$$\frac{dF}{dx}=KF$$

siendo K una constante.

Ejercicio

13.17. Dado un número de núcleos radiactivos N, realiza un programa que calcule el número de los que existen en un determinado instante, si en cada intervalo de tiempo se desintegra la mitad de ellos.

Un problema de ecología

Consideraremos aquí la interrelación entre dos especies animales, por ejemplo, conejos y zorros, donde la segunda especie se alimenta de la primera.

Si x es el número de conejos e y el número de zorros, se admite que el tamaño de las dos poblaciones cambia según las leyes:

$$\frac{dx}{dt} = Ax - Bxy$$

$$\frac{dy}{dt} = Cxy - Dy$$

Si no hay zorros, los conejos aumentan proporcionalmente a su número Ax y si no hay conejos, los zorros se morirían de acuerdo con la expresión $-Dy$. Los términos en xy reflejan la interrelación entre ambas, y representan un crecimiento en la población de zorros y una disminución en la de conejos.

En forma de incrementos finitos, las relaciones anteriores pueden escribirse así:

$$\frac{\Delta x}{\Delta t} = Ax - Bxy \qquad \frac{\Delta y}{\Delta t} = Cxy - Dy$$

Y despejando:

$$x(t+\Delta t) = (Ax(t) - Bx(t) \cdot y(t))\ \Delta t + x(t)$$

$$y(t+\Delta t) = (Cx(t) \cdot y(t) - Dy(t))\ \Delta t + y(t)$$

Traduzcamos las expresiones anteriores a un programa en BASIC que permite calcular, para cada momento, la población de conejos x y de zorros y.

Programa

```
  1 REM ECOLOGIA
 10 PRINT "INTRODUZCA LAS CONSTANTES A, B, C, D"
 20 INPUT A
 30 INPUT B
 40 INPUT C
 50 INPUT D
 60 PRINT "INTERVALO DE TIEMPO"
 70 INPUT IT
 80 PRINT "MAXIMO INTERVALO DE TIEMPO"
 90 INPUT MT
100 PRINT "NUMERO INICIAL DE CONEJOS"
110 INPUT X
120 PRINT "NUMERO INICIAL DE ZORROS"
130 INPUT Y
```

Hasta aquí se han introducido los valores de las constantes A, B, C, D, el intervalo de tiempo IT y el máximo tiempo MT, así como los números iniciales de zorros y conejos.

```
140  FOR T=0 TO MT STEP IT
150  LET X=X+(A * X-B * X * Y) * IT
160  LET Y=Y+(C * X * Y-D * Y) * IT
170  LET T=T+IT
180  PRINT T, X, Y
190  NEXT T
200  END
```

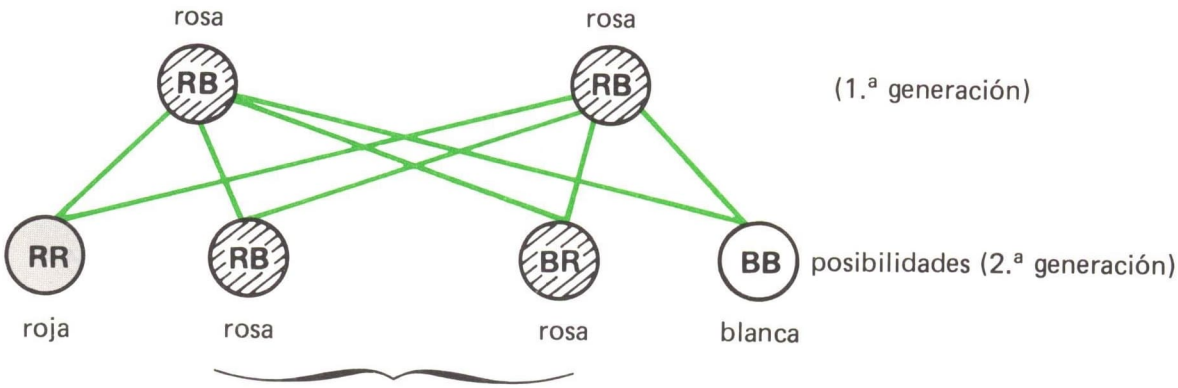
En este bucle se calculan los sucesivos valores de T, X e Y.
Se obtienen valores adecuados usando como datos:
A=4 B=2 C=1 D=3
IT=0.01 MT=4.2 tomando X=3 e Y=0.5 (3 y 0.5 no significa que se parta inicialmente con 3 conejos y 0.5 zorros, sino que la relación conejos a zorros es de 3 a 0.5).

Ejercicio

13.18. Ampliar el programa anterior para que se obtenga una gráfica del número de zorros frente al de conejos.

Estudio de leyes genéticas

Vamos a analizar una de las leyes genéticas: la segunda ley de Mendel.
Recordaremos esta ley mediante un ejemplo.
Si se cruzan flores rojas (RR) con flores blancas (BB), en la primera generación surgen flores híbridas, de color rosa (RB) (*). Al cruzarse dos flores de color rosa de la primera generación, en la segunda generación pueden surgir estas posibilidades:

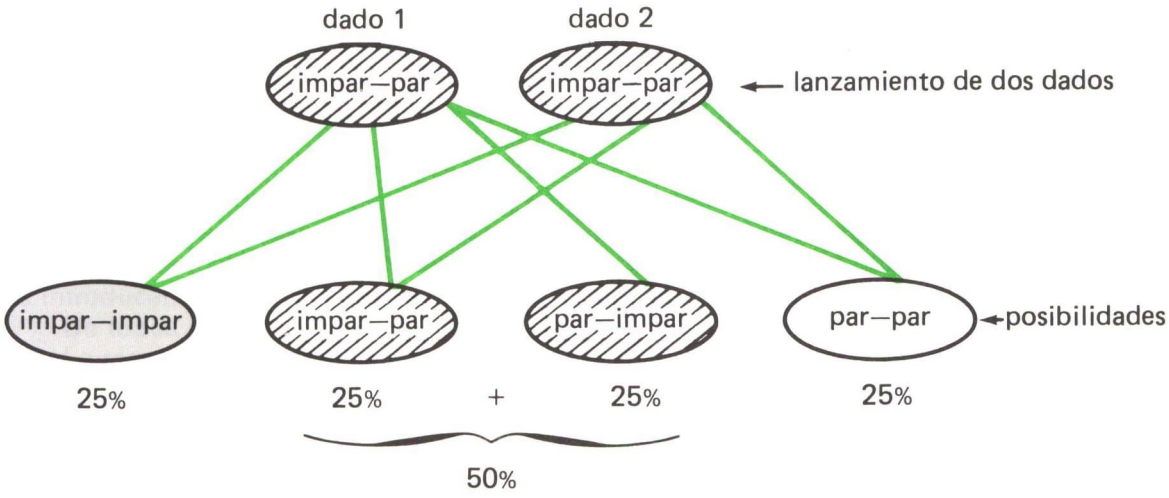


Es decir, si se parte de dos individuos híbridos de la primera generación, en la segunda generación existe 25% de posibilidades de que la flor sea roja, 50% de que sea de color rosa y 25% de que sea blanca. O, pasando a términos de frecuencia, de 100.000 individuos de esta segunda generación, unos 25.000 serán rojos, unos 50.000 serán de color rosa y unos 25.000 serán blancos.

(*) RR y BB representan genotipos de dos razas puras, cada uno de ellos con dos genes R, R y R, B. El genotipo RB es híbrido, por tener dos genes diferentes R y B.

La comprobación experimental de esta ley lleva mucho tiempo y exige mucho trabajo. Pero con ayuda del microordenador se pueden simular los experimentos y comprobar rápidamente la ley. Para ello simularemos el lanzamiento de dos dados; cada uno de ellos representa una flor híbrida, en cuanto que contiene, por decirlo así, dos genes diferentes: impar (R) y par (B).

Al lanzar los dos dados pueden presentarse cuatro posibilidades:



El programa siguiente hace la simulación.

Programa

```
 10  PRINT ''NUMERO DE LANZAMIENTOS''
 20  INPUT N
 30  LET C=0
 40  LET D=0
 50  LET E=0
 60  FOR I=1 TO N
 70  LET X=INT(RND (1) * 6+1)
 80  LET Y=INT(RND (1) * 6+1)
 90  FOR J=1 TO 3
100  FOR K=1 TO 3
110  IF X=(2 * J–1) AND Y=(2 * K–1) THEN 180
120  IF X=(2 * J–1) AND Y=2 * K THEN 200
130  IF X=2 * J AND Y=(2 * K–1) THEN 200
140  IF X=2 * J AND Y=2 * K THEN 160
150   GOTO 210
160  LET C=C+1
170  GOTO 210
180  LET D=D+1
190  GOTO 210
200  LET E=E+1
210  NEXT K
220  NEXT J
230  NEXT I
```

En los bucles anidados se lanzan los dados N veces y se obtienen los resultados X e Y de cada lanzamiento.

En los bucles correspondientes a las instrucciones 90 a 220 se cuenta el número de veces que sale cada combinación, teniendo en cuenta que la combinación pares-impares se repite.

```
240  PRINT C
250  PRINT D
260  PRINT E
270  END
```

Finalmente se imprime el resultado obtenido al contar las diversas combinaciones.

14 Geografía e Historia

Averiguación de capitales de países

El objeto del siguiente programa es comprobar ciertos conocimientos geográficos, como, por ejemplo, capitales y países, ríos y países, etc., aunque también podría utilizarse en Literatura para averiguar autores y obras, en Química para repasar símbolos y nombres de elementos químicos, etc.

En primer lugar, se introducen en la memoria, como variables alfanuméricas con índices, los países y capitales: P$(I), C$(I).

Para dar cada contestación se introduce otra variable alfanumérica, B$(I). El número máximo de letras que puede tener cada país o capital es 20, y el número N de países que se introducen varía a voluntad.

De todos los países y capitales se extraerá una al azar escogiendo su índice (S).

La respuesta se comparará con el contenido de la variable CS(I). En caso de error se dará el resultado correcto, y en caso de acierto se sumará una unidad a un contador.

Programa

```
 1  REM AVERIGUA CAPITALES
10  DIM B$(1,20)
15  INPUT N
20  DIM P$(N, 20)
25  DIM C$(N, 20)
```

En esta parte del programa se declaran las variables que se van a utilizar y se pide el número N de países y capitales (*).

```
30  FOR I=1 TO N
40  INPUT P$(I)
50  INPUT C$(I)
60  NEXT I
```

Hasta aquí se han introducido los N países y N capitales en la memoria del microordenador.

```
65  LET T=0
```

La instrucción 65 inicializa un contador.

```
70  FOR J=1 TO 10
80  LET S=INT (RND(1) * N+1)
90  PRINT «CAPITAL DE»; P$(S)
100 INPUT B$(1)
105 PRINT B$(1)
110 IF B$(1)=C$(S) THEN 200
120 PRINT ''TE EQUIVOCASTE; EL RESULTADO ES: '' ; C$(S)
130 GOTO 220
200 LET T=T+1
210 PRINT ''ACERTASTE''
220 NEXT J
```

(*) Algunos microordenadores dan mensaje de error si no se pone el segundo número.

En esta parte se obtiene un índice S al azar y se compara la capital correspondiente con la respuesta dada. Con el bucle 70-220 se repite el proceso 10 veces.

```
230  PRINT "EL NUMERO DE ACIERTOS ES"; T
240  END
```

Se imprime finalmente el número de aciertos (contenido del contador T).

Ejercicios

14.1. Aplicar el programa anterior a un conjunto concreto de capitales y estados.

14.2. Aplicar el programa anterior a un conjunto concreto de autores y sus obras literarias.

14.3. Aplicar el programa anterior a un conjunto concreto de elementos químicos y sus símbolos.

Pirámides de población

En Geografía se plantea frecuentemente el problema de la distribución de la población de un país por edades y sexo. La representación gráfica de esta distribución es lo que llamamos *pirámide de población*.

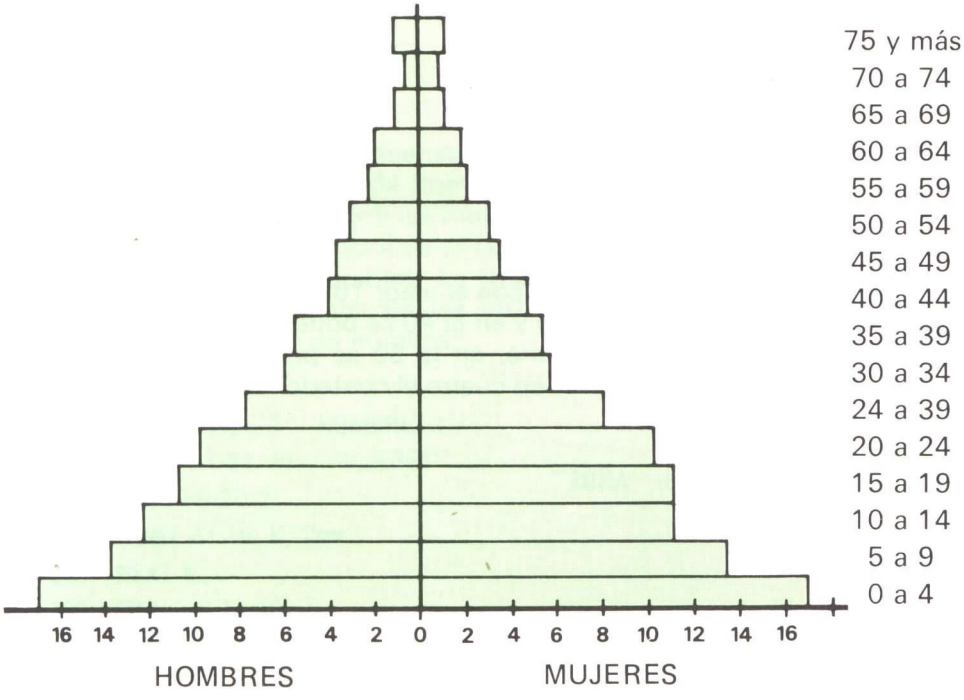
Así, por ejemplo, la población de Guatemala en el año 1960 era de 3 965 000 habitantes aproximadamente, y su distribución por edades, de cinco en cinco, era la siguiente:

Grupo de edades (años)	Número de hombres (miles)	Número de mujeres (miles)
0 a 4	321,6	344,3
5 a 9	262,6	278,3
10 a 14	232,2	233,0
15 a 19	201,7	233,0
20 a 24	182,7	212,4
25 a 29	148,4	165,0
30 a 34	114,2	117,5
35 a 39	106,2	115,5
40 a 44	76,1	90,7
45 a 49	73,3	76,3
50 a 54	56,1	63,9
55 a 59	41,9	41,2
60 a 64	38,1	39,2
65 a 69	19,0	18,6
70 a 74	11,4	14,4
75 y más	17,1	18,6

El total de hombres es igual a 1 903 000 y el total de mujeres 2 062 000. A partir de estos datos se calculan las frecuencias relativas correspondientes, dividiendo el número de hombres (o mujeres) de un grupo determinado de edades, entre el total de hombres (o mujeres). La tabla de porcentajes que se obtiene es:

Grupos de edad (años)	Frecuencia relativa (hombres)	Frecuencia relativa (mujeres)
0 a 4	0,169	0,167
5 a 0	0,138	0,135
10 a 15	0,122	0,113
15 a 19	0,106	0,113
20 a 24	0,960	0,103
25 a 29	0,780	0,800
30 a 34	0,600	0,570
35 a 39	0,560	0,560
40 a 44	0,400	0,440
45 a 49	0,380	0,370
50 a 54	0,300	0,31
55 a 59	0,220	0,20
60 a 64	0,200	0,19
65 a 69	0,100	0,09
70 a 74	0,060	0,07
75 y más	0,090	0,09

Con estos datos se puede construir la pirámide de población correspondiente, colocando en la parte izquierda los hombres y en la derecha las mujeres. Al eje vertical se ha llevado la edad de cinco en cinco años y al eje horizontal la frecuencia relativa porcentual (frecuencia relativa $\times 100$).



Programa

Los elementos necesarios para diseñar un programa que nos dé la pirámide de población son los siguientes:

1.º Dos matrices H y M, de dimensión 16 cada una (una dimensión por cada grupo de edad), para almacenar los datos de los grupos de edad de hombres y mujeres, respectivamente. Además, dos totalizadores de hombres y mujeres TH y TM.

2.º Dos bucles para ir asignando los valores de los distintos grupos de edad: primero, a la matriz H y, segundo, a la matriz M. En estos bucles se contabilizarán todos los hombres, TH, y todas las mujeres, TM.

3.º Un bucle en el que se calculen los porcentajes (H/TH y M/TM); como estos nuevos valores van a estar entre 0 y 0.20 aproximadamente, los multiplicamos por 50 para que se encuentren entre 0 y 10, pues de esta forma es más fácil representarlos gráficamente.

$$H(K)=(H(K)/TH) * 50$$
$$M(K)=(M(K)/TM) * 50$$

Por último, en este mismo bucle, estos valores son redondeados al entero más próximo. Es decir, si, por ejemplo, H(3)=5.6 tomamos H(3)=6 y si H(4)=3.5 tomamos H(4)=3.

4.º El paso siguiente consiste en dibujar los ejes y en la parte inferior izquierda poner HOMBRES y en la parte inferior derecha MUJERES.

5.º Por medio de dos bucles anidados se dibujan en la parte izquierda de la gráfica los puntos correspondientes a los hombres y, en la parte derecha, los correspondientes a las mujeres.

```
 1  REM PIRAMIDE DE POBLACION
10  DIM H(16)
20  DIM M(16)
30  LET TH=0
40  LET TM=0
50  LET I=0
60  LET J=4
```

En las instrucciones 10 y 20 se dimensionan con el valor 16 las matrices de grupos de hombres y de grupos de mujeres H y M. En la 30 y en la 40 se ponen a cero los contadores total hombres TH y total mujeres TM. Por último, en la 50 se pone a cero el contador izquierdo de los grupos de edad y la 60 inicializa en cuatro el contador derecho (0 a 4 años).

```
 70  FOR K=1 TO 16
 80  PRINT ''HOMBRES DE''; I; ''A''; J; ''AÑOS''
 90  INPUT H(K)
100  LET TH=TH+H(K)
110  LET I=I+5
120  LET J=J+5
130  NEXT K
140  CLS
```

La instrucción 80 imprime el grupo de edad que hay que introducir, introducción que se hace en la instrucción 90. En la 100 el total de hombres TH se incrementa con el valor introducido en 90. Las instrucciones 110 y 120 incrementan los límites del grupo de edad en cinco. La 130 repite el bucle hasta que se introduzcan los 16 valores. La 140 borra la pantalla.

```
150  LET I=0
160  LET J=4
170  FOR K=1 TO 16
180  PRINT ''MUJERES DE''; I; ''A''; J; ''AÑOS''
190  INPUT M(K)
200  LET TM=TM+M(K)
210  LET I=I+5
220  LET J=J+5
230  NEXT K
240  CLS
```

Las instrucciones 150 a 160 vuelven a inicializar los límites de grupos de edad I y J, para los valores de mujeres. Desde la instrucción 170 a la 230 se repite el bucle anterior, sólo que para mujeres. Se termina borrando la pantalla con la instrucción 240.

```
250  FOR K=1 TO 16
260  LET H(K)=(H(K)/TH) * 50
270  IF H(K)-INT (H(K))>0.5 THEN LET H(K)=H(K)+1
280  LET H(K)=INT (H(K))
290  LET M(K)=(M(K)/TM) * 50
300  IF M(K)-INT (M(K))>0.5 THEN LET M(K)=M(K)+1
310  LET M(K)=INT (M(K))
320  NEXT K
```

La instrucción 250 da comienzo al bucle que termina en la 320. En la 260 se calcula el porcentaje de cada grupo de hombres respecto del total de hombres; como este valor que se obtiene está entre 0 y 1, se multiplica por 50 para luego poder hacer la representación gráfica. En la 270 se observa si la parte decimal de H(K) es mayor que 0.5; en caso afirmativo, se suma una unidad. La instrucción 280 deja a H(K) con su parte entera, eliminando la parte decimal. En las instrucciones 290, 300 y 310 se realiza el mismo proceso para los grupos de mujeres.

```
330  PRINT AT 21, 7; ''HOMBRES''
340  PRINT AT 21, 17; ''MUJERES''
350  FOR K=0 TO 31
360  PRINT AT 19, K; ''▬''
370  NEXT K
380  FOR K=0 TO 18
390  PRINT AT K, 15; ''▌ ''
400  NEXT K
```

Las instrucciones 330 y 340 escriben las palabras HOMBRES y MUJERES en la parte inferior del gráfico. El bucle 350-370 dibuja el eje horizontal y el 380-400 el eje vertical.

```
410  FOR K=1 TO 16
420  FOR L=0 TO H(K)
430  PRINT AT 19-K, 15-L; "■"
440  NEXT L
450  NEXT K
```

Este doble bucle dibuja la parte de la pirámide de población correspondiente a los 16 grupos de edades de hombres.

```
460  FOR K=1 TO 16
470  FOR L=0 TO M(K)
480  PRINT AT 19-K, 15+L; "■"
490  NEXT L
500  NEXT K
```

Análogamente al bloque anterior, éste dibuja la pirámide correspondiente a los 16 grupos de edades de mujeres.

Ordenar cronológicamente

El siguiente programa sirve para comprobar el acierto en una ordenación cronológica de acontecimientos. Serviría también en Literatura para situar en el tiempo a ciertos autores, así como para comprobar la ordenación de elementos según energías de ionización, radios atómicos, etc.

Inicialmente se introducen estos acontecimientos ordenadamente en la memoria. El número de acontecimientos introducidos N puede determinarse a voluntad, no pudiendo exceder de 50 letras (aunque este número puede cambiarse). El número de acontecimientos que salen en pantalla al azar es tres, distintos entre sí.

Para evitar escribir demasiado, se colocan las letras A, B y C delante de ellos y éstas son las respuestas que deben darse.

Las respuestas se comparan con el contenido de la memoria y, en caso de error, el programa da la respuesta correcta.

Programa

```
10  INPUT N
20  DIM A$(N, 50)
30  DIM C(3)
40  DIM D$(3,50)
```

Se declaran variables que se van a utilizar y se introduce el número de acontecimientos.

```
50  FOR I=1 TO N
60  INPUT A$(I)
70  NEXT I
```

Con este bucle se introducen ordenadamente los distintos acontecimientos

```
 72  LET D$(1)=''A''
 74  LET D$(2)=''B''
 76  LET D$(3)=''C''
```

Se almacenan los valores A, B y C en D$(1), D$(2), D$(3) que después se compararán con las contestaciones.

```
 90  LET S=INT (RND(1) * N+1)
100  LET T=INT (RND(1) * N+1)
105  IF T=S THEN 100
110  LET X=INT (RND(1) * N+1)
115  IF X=S OR X=T THEN 110
```

Se eligen tres respuestas al azar que sean distintas

```
120  PRINT ''A''; A$(S)
130  PRINT ''B''; A$(T)
140  PRINT ''C''; A$(X)
```

Se imprimen los tres acontecimientos elegidos con las letras A, B y C delante.

```
150  FOR J=1 TO 3
160  INPUT B$(J)
170  PRINT B$(J)
180  IF B$(J)=D$(1) THEN 220
190  IF B$(J)=D$(2) THEN 240
200  IF B$(J)=D$(3) THEN 260
210  GOTO 550
220  LET B$(J)=A$(S)
230  GOTO 270
240  LET B$(J)=A$(T)
250  GOTO 270
260  LET B$(J)=A$(X)
270  NEXT J
```

Pide una contestación A, B o C y después se asigna la contestación A$(S), A$(T) o A$(X) a B$(1), B$(2) y B$(3), respectivamente.

```
280  LET C(1)=S
290  LET C(2)=T
300  LET C(3)=X
310  FOR M=1 TO 2
320  FOR N=1 TO 2
330  LET X=C(N)
340  LET Y=C(N+1)
350  IF X<=Y THEN 380
360  LET C(N)=Y
370  LET C(N+1)=X
380  NEXT N
390  NEXT M
```

Ordena S, T y X en orden creciente (ver programa de ordenación de números).

```
400  LET S=C(1)
410  LET T=C(2)
420  LET X=C(3)
```

Coloca en S, T y X el valor correctamente ordenado.

```
430  IF B$(1)=A$(S) THEN 450
440  GOTO 500
450  IF B$(2)=A$(T) THEN 470
460  GOTO 500
470  IF B$(3)=A$(X) THEN 490
480  GOTO 500
490  PRINT ''ACERTASTE''
495  GOTO 90
500  PRINT ''TE EQUIVOCASTE, EL RESULTADO ES:''
510  PRINT A$(S)
520  PRINT A$(T)
530  PRINT A$(X)
540  GOTO 90
550  END
```

Esta parte del programa compara las respuestas con el contenido de la memoria y en caso de error da la respuesta correcta.

El programa podría mejorarse borrando la pantalla (CLS) entre las instrucciones 490 y 495 estableciendo un tiempo de espera (con PAUSE, WAIT, o instrucción similar según el ordenador) y lo mismo entre 530 y 540 (CLS y PAUSE, WAIT).

Ejercicios

14.4. Aplica el programa anterior a una serie concreta de acontecimientos históricos.

14.5. Aplica el programa anterior para situar en el tiempo a un conjunto concreto de autores literarios.

14.6. Aplica el programa anterior para ordenar en orden creciente de radios atómicos una serie de elementos.

14.7. Aplica el programa anterior para ordenar en orden creciente de potenciales de ionización una serie de elementos.

15 Lengua e idiomas

Ejercicio de acentuación

Este programa da un texto literario de la obra *Platero y yo*, de Juan Ramón Jiménez, al que le faltan los acentos. El alumno debe escribir las palabras que están mal escritas.

El texto es el siguiente:

«ACARICIE A PLATERO Y, COMO PUDE, LO ENGANCHE A LA CARRETILLA, DELANTE DEL BORRICO MISERABLE, LO OBLIGUE ENTONCES, CON UN CARIÑOSO IMPERIO, Y PLATERO, DE UN TIRON, SACO CARRETILLA Y RUCIO DEL ATOLLADERO, Y LES SUBIO LA CUESTA.»

Las palabras solución son:

ACARICIÉ, ENGANCHÉ, OBLIGUÉ, TIRÓN, SACÓ, SUBIÓ

Lo primero que hay que hacer es almacenar el texto en memoria mediante una subrutina, para que aparezca en pantalla todas las veces que sea necesario. Las palabras solución se almacenan en una matriz B$ de dimensión 6×8, pues hay seis palabras y la longitud máxima de ellas es ocho.

Se define una variable A$ para que el alumno dé sus respuestas prolongando su longitud hasta ocho, añadiendo blancos si su longitud es inferior a ese número, a fin de que pueda ser comparada con la solución B$.

Por último, se hace una comparación entre la respuesta A$ y la solución B$. Si es correcta, se pasa a la siguiente solución y, en caso contrario, se obliga a que el alumno responda otra vez. Este proceso se repite hasta que se conteste correctamente todas las palabras, con lo que el programa se termina.

Programa

```
 1  REM ACENTUACION
10  PRINT ''DEBES DAR LAS PALABRAS QUE LLEVEN ACENTO, DE UNA EN UNA''
20  DIM B$ (6,8).
30  LET B$ (1)=''ACARICIE''
40  LET B$ (2)=''ENGANCHE''
50  LET B$ (3)=''OBLIGUE''
60  LET B$ (4)=''TIRON''
70  LET B$ (5)=''SACO''
80  LET B$ (6)=''SUBIO''
```

En la instrucción 10 aparece lo que el alumno debe hacer durante la ejecución del programa. La instrucción 20 da la dimensión de 6 cadenas de a lo más 8 caracteres. Por último, de la 30 a la 80 se introducen en la matriz B$ las soluciones del problema.

```
 90  FOR I=1 TO 6
100  GOSUB 500
110  INPUT A$
120  LET L=LEN (A$)
130  FOR J=L+1 TO 8
140  LET A$=A$+" "
150  NEXT J
```

La instrucción 90 abre un bucle para las seis respuestas válidas. La 100 envía a la subrutina; ésta escribe el texto que se debe acentuar. La 110 sirve para que el alumno introduzca su respuesta. Como esta respuesta A$ puede ser de longitud inferior a 8 (menos de ocho letras), es necesario calcular su longitud (instrucción 120), y posteriormente añadir blancos, al final de dicha palabra, hasta alcanzar la longitud 8; esto se consigue con el bucle que comprende las instrucciones 130, 140 y 150.

```
160  IF A$=B$ (I) THEN 190
170  PRINT "ERROR; PRUEBA DE NUEVO"
180  GOTO 110
```

La instrucción 160 compara la respuesta del alumno A$ con la solución, B$ (I); si no es correcta, pasa el control a la instrucción siguiente (instrucción 170), la cual le anuncia que es errónea la respuesta y que pruebe de nuevo; por ello, la instrucción 180 envía el control del programa a la 110, para que el alumno introduzca otra respuesta (hasta que no sea correcta no se sale de este bucle). Si la respuesta es la correcta, la instrucción 160 envía el control a la 190.

```
190  CLS
200  PRINT "CORRECTO, BUSCA LA PALABRA SIGUIENTE"
210  NEXT I
220  PRINT "NO BUSQUES MAS; HAS ENCONTRADO TODAS"
230  STOP
```

La 190 borra toda la pantalla y la 200 anuncia que la respuesta es correcta y que el alumno se prepare para la siguiente. En la 210 se termina el bucle comenzado en la instrucción 90.

El fin del ejercicio se anuncia con la instrucción 220 y el programa termina con la instrucción 230 STOP para que no se ejecuten las instrucciones siguientes, correspondientes a la subrutina.

```
500  PRINT "ACARICIE A PLATERO Y, COMO PUDE, LO ENGANCHE A LA CARRETILLA, DELANTE DEL
     BORRICO MISERABLE, LE OBLIGUE ENTONCES, CON UN CARIÑOSO IMPERIO, Y PLATERO, DE UN
     TIRON, SACO CARRETILLA Y RUCIO DEL ATOLLADERO, Y LES SUBIO LA CUESTA"
510  RETURN
```

Estas dos últimas instrucciones componen la subrutina, llamada desde la instrucción 100. La 500 escribe el texto de estudio, que se repetirá 6 veces a lo largo de la ejecución del programa (una para cada respuesta), y la 510 envía el control a la instrucción siguiente: a la 100.

Ejercicios

15.1 Realmente en este programa no es necesario el uso de la subrutina pues sólo se utiliza en un punto del programa. Eliminar la subrutina introduciendo sus instrucciones en el punto correspondiente del programa.

15.2 Generalizar el programa para cualquier texto y para un número indeterminado de palabras acentuadas.

Conjugación de verbos

Los verbos regulares se dividen en tres grupos, atendiendo a su terminación; en AR, en ER y en IR. La conjugación de todos los verbos de un mismo grupo es bastante sencilla pues sus terminaciones (desinencias) son iguales. Así, por ejemplo, para conjugar el presente de indicativo del verbo TARDAR basta con añadir a la parte radical TARD las terminaciones

O
AS
A
AMOS
AIS
AN

y anteponer los pronombres personales correspondientes

YO	TARD	O
TU	TARD	AS
EL	TARD	A
NOSOTROS	TARD	AMOS
VOSOTROS	TARD	AIS
ELLOS	TARD	AN

El programa siguiente da el presente de indicativo, el pretérito imperfecto o co-pretérito, el pretérito indefinido o pretérito y el futuro imperfecto o futuro (todos estos tiempos forman el modo indicativo simple de un verbo de la primera conjugación, o sea, de los terminados en AR).

Lo primero que hace el programa es pedir el verbo y almacenarlo en A$. A continuación calcula la longitud de A$ y extrae una subcadena B$, en la que falte la terminación AR.

Por último, se imprimen los distintos tiempos del modo indicativo simple.

Programa

```
 1  REM MODO INDICATIVO SIMPLE DE LA PRIMERA CONJUGACION
10  INPUT "VERBO", A$
20  LET L=LEN (A$)
30  LET B$=LEFT$ (A$; L-2)
```

En la instrucción 10 se introduce el verbo, en la 20 se calcula su longitud y en la 30 se extrae una subcadena izquierda de longitud L−2. Luego B$ es la parte radical del verbo.

```
 40 PRINT "PRESENTE DE INDICATIVO"
 50 PRINT "YO", B$; "O"
 60 PRINT "TU", B$; "AS"
 70 PRINT "EL", B$; "A"
 80 PRINT "NOSOTROS", B$; "AMOS"
 90 PRINT "VOSOTROS", B$; "AIS"
100 PRINT "ELLOS", B$; "AN"
```

El bloque 40-100 imprime el presente de indicativo. Por ejemplo, la instrucción 50 imprime YO, salta a la siguiente zona (debido a la coma) y escribe B$ junto a 0, pues al estar separados por punto y coma se imprimen seguidos.

```
110 PRINT "PRETERITO IMPERFECTO O CO-PRETERITO"
120 PRINT "YO", B$; "ABA"
130 PRINT "TU", B$; "ABAS"
140 PRINT "EL", B$; "ABA"
150 PRINT "NOSOTROS", B$; "ABAMOS"
160 PRINT "VOSOTROS", B$; "ABAIS"
170 PRINT "ELLOS", B$; "ABAN"
```

Este bloque imprime el pretérito imperfecto o co-pretérito.

```
180 PRINT "PRETERITO INDEFINIDO O PRETERITO"
190 PRINT "YO", B$; "E"
200 PRINT "TU", B$; "ASTE"
210 PRINT "EL", B$; "O"
220 PRINT "NOSOTROS", B$; "AMOS"
230 PRINT "VOSOTROS", B$; "ASTEIS"
240 PRINT "ELLOS", B$; "ARON"
```

Las instrucciones 180-240 imprimen el pretérito indefinido o pretérito.

```
250 PRINT "FUTURO IMPERFECTO O FUTURO"
260 PRINT "YO", A$; "E"
270 PRINT "TU", A$; "AS"
280 PRINT "EL", A$; "A"
290 PRINT "NOSOTROS", A$; "EMOS"
300 PRINT "VOSOTROS", A$; "EIS"
310 PRINT "ELLOS", A$; "AN"
```

En el bloque 250-310 se toma como parte radical el verbo completo A$, ya que la conjugación del futuro incluye en la parte radical la terminación AR.

Ejercicios

15.3. Ampliar el programa anterior con los modos potencial o condicional, subjuntivo e imperativo.

15.4. Diseñar un programa en el cual las desinencias y los pronombres personales se almacenen precisamente en tablas para después utilizar bucles en la impresión.

15.5 Hacer un programa que introduzca el verbo, reconozca si es de la primera, segunda o tercera conjugación, pregunte qué tiempo es el que interesa y a continuación lo conjugue.

Identificación de preposiciones inglesas

El planteamiento del problema consiste en que vayan apareciendo en pantalla una serie de frases en inglés, a las que le faltan las preposiciones. El que ejecute el programa deberá dar una preposición (respuesta) y la pantalla le comunicará si es correcta o no. Si lo es, pasará a la siguiente frase; en caso contrario, volverá a aparecer en la pantalla la misma frase.

Las frases son	*Las preposiciones son*
JOHN DREAMT ... MARY LAST NIGHT	OF
I'M LOOKING FORWARD ... MEETING YOU NEXT WEEKEND	TO
MARY IS THINKING ... JOHN AT THE MOMENT	OF
DON'T STAND THERE ... THE DOOR, COME ... AND SIT ...	AT-IN-DOWN
YOU CAN COUNT ... ME, I NEVER LET MY FRIENDS ...	ON-DOWN

El programa es abierto, es decir, sirve para cualquier número N de frases (en este caso N=5) y no únicamente en inglés sino en cualquier otro idioma; basta con dar las frases y preposiciones en el idioma elegido.

La entrada de los datos, frases y preposiciones, se realiza por medio de un bucle que tiene la instrucción READ A$, B$, almacenándose en A$ las frases y en B$ las preposiciones.

Previamente hay que dimensionar las matrices A$ y B$. Se supone que el número máximo de letras para las frases es 100, y 20 para las preposiciones.

El paso siguiente es definir dos variables Y$ y Z$ que, por medio de su impresión en pantalla, informen si la respuesta elegida es correcta o no.

También es necesario añadir a las respuestas X$ blancos, para que las palabras alcancen la longitud 20 y puedan ser comparadas con las soluciones B$.

Por último, el programa ha de contener un bucle en el que se dé una de las frases, se introduzca una respuesta y ésta sea comparada con la solución. No se pasará a la frase siguiente hasta que se acierte la respuesta.

Programa

```
 1  REM PREPOSICIONES EN INGLES
10  INPUT N
20  DIM A$ (N, 100), B$ (N, 20)
30  FOR I=1 TO N
40  READ A$ (I), B$ (I)
50  NEXT I
60  LET Y$="NO ES CORRECTA, REPITE"
70  LET Z$="CORRECTA, AQUI TIENES OTRA FRASE"
```

En la instrucción 10 se introduce el número de frases que hay. La 20 dimensiona las N frases y las N preposiciones con un máximo de 100 y 20 caracteres, respectivamente. De la 30 a la 50, el bucle lee las frases y las preposiciones. Por último, en la 60 y 70 se definen dos variables, Y$ y X$, que servirán para decir al alumno si ha acertado o no.

```
 80 FOR I=1 TO N
 90 PRINT ''PON LA PREPOSICION QUE FALTA Y SI HAY VARIAS DEJA UN ESPACIO ENTRE ELLAS''
100 PRINT A$ (I)
110 INPUT X$
120 LET L=LEN (X$)
130 FOR J=L+1 TO 20
140 LET X$=X$+'' ''
150 NEXT J
```

En la instrucción 80 comienza la serie de instrucciones que llevarán al alumno a contestar las distintas cuestiones. En la 90 se le dice qué es lo que debe responder y cómo, en el caso de varias preposiciones. La instrucción 100 le da la frase que tiene que completar con la preposición o preposiciones correctas. Por último, en la 110 se le pide la respuesta X$, se calcula su longitud L en la 120 y en el bucle 130-150 se añaden a su respuesta X$ los blancos necesarios para que la cadena alcance la longitud 20 y así puede ser comparada con la solución B$.

```
160 IF X$=B$ (I) THEN 190
170 PRINT Y$
180 GOTO 100
```

La instrucción 160 compara la solución B$ con la respuesta X$ dada por el alumno; si no es correcta, la 170 le dice que repita, por medio de la variable Y$ definida en 60; y la 180 devuelve el control a la instrucción 100, con lo que se da la oportunidad al alumno de repetir su respuesta X$. Este proceso no termina mientras no dé la respuesta correcta.

```
190 CLS
200 PRINT Z$
210 NEXT I
220 PRINT ''¡ENHORABUENA! LLEGASTE A LA META''
```

A la instrucción 190 se llega cuando el alumno ha acertado la preposición correspondiente; esta instrucción borra toda la pantalla. Posteriormente, se da al alumno, por medio de la variable Z$ (en la instrucción 200), la información de haber acertado con su respuesta. Mediante la instrucción 210 se repite el ciclo con otra frase. Con la instrucción 220 se le comunica al alumno el fin de las frases y, por tanto, la terminación del programa.

Este programa sirve para cualquier número de frases, pero si se quiere utilizar para el ejemplo presentado, hay que introducir los siguientes datos:

```
230 DATA 5
240 DATA ''JOHN DREAMT ... MARY LAST NIGHT'', ''OF''
250 DATA ''I'M LOOKING FORWARD ... MEETING YOU NEXT WEEKEND'', ''TO''
260 DATA ''MARY IS THINKING ... JOHN AT THE MOMENT'', ''OF''
270 DATA ''DON'T STAND THERE ... THE DOOR, COME ... AND SIT ...'', ''AT IN DOWN''
280 DATA ''YOU CAN COUNT ... ME, I NEVER LET MY FRIENDS ...'', ''ON DOWN''
290 END
```

Ejercicios

15.6. En el programa ejemplo poner un contador, de tal forma que, si se cometen tres fallos en una misma frase, se pase a la siguiente.

15.7. Además del contador, poner un totalizador de fallos e imprimir su valor al final del programa.

15.8 Cambiar los datos por frases en castellano que también les falten las preposiciones.

Repasando la morfología y sintaxis latina

En el texto siguiente:

«ACEPTA ORATIONE EORUM, CAESAR OBSIDES TRADI IMPERAT EOSQUE ADUCCI IUBET.
HIS ADDUCTIS, UT IMPERAVERAT, ARBITROS INTER CIVITATES DAT, QUI LITEM AESTIMENT.»

el alumno debe buscar los sustantivos (excluidos los nombres propios) y, una vez encontrados todos, ponerlos en el mismo caso pero en número distinto al que tienen en el texto.

Las soluciones son:

ORATIONE ⟶ ORATIONIBUS
OBSIDES ⟶ OBSIDEM
ARBITROS ⟶ ARBITRUM
CIVITATES ⟶ CIVITATEM
LITEM ⟶ LITES

(CAESAR ES NOMBRE PROPIO)

Se utilizan dos variables B$ (I) y C$ (I) para introducir en la memoria del microordenador las palabras solución.

El texto, incluido en dos variables T$ y U$, se almacena en una subrutina para no sobrepasar el límite de 255 caracteres.

En un bucle se comprueba si cada respuesta es correcta. Cuando se han encontrado *todos* los sustantivos, se pide que se pongan en distinto número. De nuevo, las respuestas se comparan con las palabras solución y en caso de equivocación se da el resultado correcto.

Programa

```
 1  REM LATIN
10  READ N
20  DIM B$ (N)
25  DIM C$ (N)
30  FOR I=1 TO N
40  READ B$ (I)
50  READ C$ (I)
60  NEXT I
```

En estas instrucciones (en combinación con las instrucciones DATA del final del programa) se introducen los sustantivos en distintos números C$ (I).

```
 70  GOSUB 500
 80  PRINT ''BUSCA TODOS LOS SUSTANTIVOS EN EL TEXTO SIGUIENTE''
 90  INPUT AS (1)
100  FOR I=1 TO N
110  IF A$ (1)=B$ (I) THEN 130
120  GOTO 160
130  PRINT ''ACERTASTE''
140  LET C=C+1
150  GOTO 190
160  NEXT I
170  PRINT ''TE EQUIVOCASTE, BUSCA DE NUEVO''
180  GOTO 90
190  IF C < N THEN 200
200  PRINT ''YA HAS ENCONTRADO TODOS''
```

Una vez puesto el texto en pantalla (instrucción 70 GOSUB 500) se piden los sustantivos (instrucciones 80 y 90) y en el bucle siguiente (instrucciones 100 a 160) se comprueba si la palabra dada es alguna de las almacenadas en la tabla B$ (I); si no es, se pide una nueva respuesta, y si es, se suma uno al contador C. Cuando C coincide con el número de sustantivos existentes (instrucciones 160 a 170) se imprime un mensaje.

```
210  WAIT (500)
220  CLS
230  PRINT ''PON LOS SUSTANTIVOS ANTERIORES EN EL MISMO CASO PERO EN DISTINTO NUMERO''
240  FOR I=1 TO N
250  PRINT B$ (I)
260  INPUT D$ (I)
270  IF D$ (I)=C$ (I) THEN 300
280  PRINT ''TE EQUIVOCASTE, LA RESPUESTA ES:''; C$ (I)
290  GOTO 310
300  PRINT ''DE ACUERDO''
310  NEXT I
320  STOP
```

Después de un tiempo de espera (instrucción 210), se borra la pantalla (instrucción 220) y se hace una nueva pregunta (230). La respuesta se compara con el contenido de la memoria C$ (I).

```
500  LET T$=''ACCEPTA ORATIONE EORUM, CAESAR OBSIDES TRADI IMPERAT EOSQUE ADUCCI IUBET''
510  LET U$=''HIS ADDUCTIS, UT IMPERAVERAT, ARBITROS INTER CIVITATES DAT, QUI LITEM
     AESTIMENT''
520  PRINT T$
530  PRINT U$
540  RETURN
```

Con la subrutina (500-540) se almacena el texto en dos variables, T$ y U$, y a continuación se imprime

```
590 DATA 5
600 DATA ORATIONE, ORATIONIBUS, OBSIDES, OBSIDEM, ARBITROS, ARBITRUM
610 DATA CIVITATES, CIVITATEM, LITEM, LITES
620 END
```

En las instrucciones DATA se incluyen las distintas soluciones.

Ejercicio

15.9. Ampliar el programa anterior con la siguiente pregunta:
Buscar los pronombres que hay en el texto y poner el acusativo singular de cada uno de ellos en los tres géneros (masculino, femenino y neutro).

Soluciones

EORUM ——→ EUM, EAM, ID
EOSQUE ——→ EUM, EAM, ID
HIS ——→ HUNC, HANC, HOC
QUI ——→ QUEM, QUOM, QUOD

16 Otras aplicaciones

Utilización de una pequeña biblioteca

Aunque el almacenamiento y manejo de gran cantidad de información se hace más adecuadamente utilizando disquetes, supondremos que no disponemos de ellos, y que sólo podemos utilizar como dispositivo de memoria externa una casete corriente.

En este supuesto, el programa que se expondrá a continuación sirve para manejar una biblioteca no excesivamente grande, realizando las siguientes funciones.

A) Almacena en memoria un conjunto de libros teniendo en cuenta la materia, submateria, autor y título de cada obra.

B) Ordena estos libros por materias y, dentro de cada materia, en orden alfabético de autores y títulos (para un mismo autor ordena los libros alfabéticamente).

C) Según se desee:

1.º Proporciona todos los libros que existen en la biblioteca sobre una materia.

2.º Proporciona los libros que existen de un autor.

3.º Proporciona la materia y el autor de un título del que se dispone en la biblioteca.

Para construir el programa se define una variable L$ (I) mediante la concatenación de la variable A$ (I) que incluye un número clave de dos cifras, con la variable B$ (I) que incluye un número en clave de una sola cifra, con la variable C$ (I) que incluye el autor de la obra (para lo que se reservan cuarenta caracteres) y con la variable D$ (I) para el título del libro en la que también se utilizan cuarenta caracteres.

Una vez ordenados los libros introducidos, tomando diversos fragmentos de la variable L$ (I), con la ayuda de las instrucciones LEFT$ ( ), MID$ ( ) y RIGHT$ ( ) se podrá localizar materia, autor o título, según se desee.

Programa

```
  1  REM BIBLIOTECA
  5  READ N
 10  DIM L $ (N)
 20  DIM A $ (N)
 25  DIM B $ (N)
 30  DIM C $ (N)
 35  DIM C(N)
 40  DIM D $ (N)
 45  DIM D(N)
 50  FOR I=1 TO N
 60  READ A $ (I)
 70  READ B $ (I)
 80  READ C $ (I)
 90  READ D $ (I)
 95  LET C (I)=LEN (C $ (I))
100  LET D (I)=LEN (D $ (I))
110  FOR J=C (I)+1 TO 40
120  LET C $ (I)=C $ (I)+" "
130  NEXT J
140  FOR K=D (I)+1 TO 40
150  LET D $ (I)=D $ (I)+" "
160  NEXT K
170  LET L $ (I)=A $ (I)+B $ (I)+C $ (I)+D $ (I)
180  NEXT I
185  DATA (DEBE DARSE N: NUMEROS DE LIBROS)
190  DATA 21, 6, BAUDELAIRE, LAS FLORES DEL MAL
200  DATA 21, 2, GARCIA MARQUEZ, EL CORONEL NO TIENE QUIEN LE ESCRIBA
```

. . . . . .

. . . . . .

. . . . . .

En esta parte se introducen la clave de materia en A$ (I) (dos caracteres), la de submateria B$(I) (un carácter), y las de autor C$(I) y título D$(I) (que no pueden superar los 40 caracteres). Para estas dos últimas variables se determina la longitud (instrucciones 95 y 100) y se añaden, mediante los bucles 110-130 y 140-160, el número de blancos necesarios para completar los 40 caracteres.

En la instrucción 170 se construye la variable L$(I) que se manejará posteriormente.

Las instrucciones DATA que siguen a la primera incluyen el número total de libros que se van a introducir; éstas deben contener un número de dos cifras que indique la materia, y una cifra para la submateria, autor y título, separados por comas.

```
1000  FOR J=1 TO N-1
1010  FOR K=J+1 TO N
1020  IF L $ (J)<L $ (K) THEN 1060
1030  LET X $=L $ (J)
1040  LET L $ (J)=L $ (K)
1050  LET L $ (K)=X $
1060  NEXT K
1070  NEXT J
```

Este conjunto de instrucciones ordena la variable L$(I) por materias, submaterias, orden alfabético de autores y, finalmente, por orden alfabético de libros (ver página 74).

```
1080  PRINT ''PARA PEDIR LOS LIBROS DE UNA MATERIA ESCRIBA 1''
1090  PRINT ''PARA PEDIR LOS LIBROS DE UN AUTOR ESCRIBA 2''
1100  PRINT ''PARA PEDIR LA MATERIA Y EL AUTOR DE UN TITULO ESCRIBA 3''
1110  INPUT P
1120  IF P=1 THEN 1200
1130  IF P=2 THEN 1300
1140  IF P=3 THEN 1500
1150  PRINT ''EL PROGRAMA NO HACE ESO''
1170  GOTO 1080
```

En estas instrucciones se elige el trabajo que se va a realizar.

```
1200  PRINT ''INTRODUZCA LA CLAVE DE UNA MATERIA''
1210  INPUT H $
1220  FOR I=1 TO N
1230  IF H $=LEFT$ (L $(I), 2) THEN 1250
1240  GOTO 1260
1250  PRINT L $ (I)
1260  NEXT I
1270  GOTO 1080
```

Comparando H $ con las dos primeras cifras de L $ (I) se localizan todos los libros de una materia (bucle 1220 a 1260).

```
1300  PRINT ''INTRODUZCA EL NOMBRE DEL AUTOR''
1310  INPUT N $
1320  LET U=LEN (N$)
1330  FOR $=U+1 TO 40
1340  LET N $=N $ +'' ''
1350  NEXT $
1355  FOR I=1 TO N
1360  IF MID$ (L $ (I), 4, 40)=N $ THEN 1380
1370  GOTO 1390
1380  PRINT L $ (I)
1390  NEXT I
1400  GOTO 1080
```

En esta parte del programa se introduce el nombre de un autor el que se completa con blancos hasta alcanzar 40 caracteres (instrucciones 1320 a 1350); a continuación, mediante el bucle 1355-1390, se compara con la parte de la variable L $ (I) correspondiente al autor.

```
1500  PRINT ''INTRODUCE UN T!TULO''
1510  INPUT T$
1520  LET V=LEN (T$)
1530  FOR R=V+1 TO 40
1540  LET T$=T$+'' ''
1550  NEXT R
1560  FOR I=1 TO N
1570  IF T$=RIGHT$ (L $ (I), 40) THEN 1590
1580  GOTO 1600
1590  PRINT L $ (I)
1600  NEXT I
1605  GOTO 1080
1610  END
```

Aquí se introduce el título de un libro y se completa con blancos hasta alcanzar 40 caracteres (1520-1550); a continuación, mediante el bucle 1560-1600, se compara con la parte correspondiente al título de L $ (I).

Ejercicio

16.1. Completar el programa anterior para que pueda llevar un registro de libros prestados y de personas a las que se han prestado estos libros.

Control de la situación académica de alumnos

El programa que se expone a continuación cumple los siguientes objetivos:

A) Almacena en memoria los apellidos y nombres de un grupo de alumnos.

B) Ordena estos apellidos y nombres alfabéticamente.

C) Almacena en memoria las asignaturas que se imparten en un determinado curso.

D) Permite introducir las calificaciones (MD, IS, SF, BI, NT, SB) y actitudes (A, B, C, D, E) de los alumnos, en cada asignatura y en cada evaluación.

E) Dados los apellidos y nombre de un alumno del grupo, permite acceder a sus calificaciones en todas las asignaturas.

F) Dados los apellidos y nombre de un alumno y el nombre de una asignatura, permite acceder a las calificaciones de ese alumno y en esa asignatura.

En el programa se utiliza:

a) Una variable A$(I) para almacenar los apellidos y el nombre de los alumnos.

b) Una variable Z$(J) para almacenar los nombre de las asignaturas.

c) Una variable N$ (I, J) para almacenar las calificaciones en las evaluaciones. Esta variable se construye concatenando las variables D $ (I, J), E $ (I, J) y F $ (I, J), que se van añadiendo en cada evaluación (aquí se suponen tres evaluaciones). Cada una de estas variables incluye tres caracteres, dos para calificar los conocimientos y uno para calificar la actitud.

Programa

```
 1  REM NOTAS
 5  READ N
10  DIM A $ (N)
20  FOR I=1 TO N
30  READ A $ (I)
40  NEXT I
50  DATA (N: número de alumnos)
51  DATA LOPEZ RODRIGUEZ FRANCISCO, ORTEGA MUÑOZ ISABEL
52  DATA
```

. . . . . .

En esta parte del programa se leen e introducen en memoria el número de alumnos del grupo y los apellidos y el nombre de cada uno de ellos.

```
100  FOR J=1 TO N-1
110  FOR K=J+1 TO N
120  IF A $ (J)<A $ (K) THEN 160
130  LET X $=A $ (J)
140  LET A $ (J)=A $ (K)
150  LET A$ (K)=X$
160  NEXT K
170  NEXT J
```

Con estas instrucciones se consigue ordenar alfabéticamente apellidos y nombres de los alumnos.

```
200  READ M
210  DIM Z$(M)
220  FOR J=1 TO M
230  READ Z$(J)
240  NEXT J
250  DATA (M: número de asignaturas)
260  DATA LITERATURA, LATIN, LENGUA EXTRANJERA
```

. . . . . .

En este grupo de instrucciones se utiliza para introducir en la memoria del microordenador el número de asignaturas y sus nombres.

```
300  REM EVALUACION 1
310  DIM N $ (N, M)
320  DIM D $ (N, M)
330  FOR I=1 TO N
340  FOR J=1 TO M
350  READ D$(I)
360  LET N$(I, J)=D$ (I, J)
370  NEXT J
380  NEXT I
390  DATA SFC, ISF, NTB
```

. . . . . .
. . . . . .
. . . . . .

```
500  REM EVALUACION 2
510  DIM E$ (N, M)
520  FOR I=1 TO N
530  FOR J=1 TO M
540  READ E$ (I)
550  LET N$ (I, J)=N$ (I, J)+'' ''+E$ (I, J)
560  NEXT J
570  NEXT I
580  DATA ISF, SFC, ...
```

. . . . . .
. . . . . .
. . . . . .

```
700  REM EVALUACION 3
710  DIM F$(N, M)
720  FOR I=1 TO N
730  FOR J=1 TO M
740  READ F$ (I, J)
750  LET N$(I, J)=N$(I, J)+'' ''+F$(I, J)
760  NEXT J
770  NEXT I
780  DATA NTB, SFC, ...
```

. . . . . .
. . . . . .
. . . . . .

En estas instrucciones se van leyendo, para cada evaluación, las notas de cada alumno en todas las asignaturas, lo que deberá tenerse en cuenta en las instrucciones DATA.

Las variables N$(I, J) van utilizándose como almacén de notas, separando las distintas evaluaciones con espacios en blanco (instrucciones 550, 750).

El número de tres evaluaciones se ha elegido por comodidad, pero el procedimiento puede ampliarse a cualquier número, introduciéndose variables G$(I, J), H$(I, J)... y repitiendo el proceso.

```
900  PRINT "SI QUIERE SABER TODAS LAS NOTAS DE UN ALUMNO ESCRIBA 1"
910  PRINT "SI QUIERE SABER LAS NOTAS DE UN ALUMNO EN UNA ASIGNATURA ESCRIBA 2"
920  INPUT P
930  IF P=1 THEN 1000
940  IF P=2 THEN 1200
950  GOTO 920
```

Aquí se ha elegido la tarea que se va a realizar a continuación:

```
1000  PRINT "APELLIDOS Y NOMBRE"
1010  INPUT U $
1020  FOR I=1 TO N
1030  IF U$=A$(I) THEN 1050
1040  GOTO 1100
1050  PRINT A$(I)
1060  FOR J=1 TO M
1070  PRINT Z$(J)
1080  PRINT N$(I, J)
1090  NEXT J
1095  GOTO 900
1100  NEXT I
```

Introduciendo los apellidos y nombre, se comparan con los incluidos en A$(I); una vez encontrado el nombre del alumno, se imprimen los nombres de las asignaturas y las calificaciones correspondientes al alumno en dichas asignaturas.

```
1200  PRINT "APELLIDOS Y NOMBRE"
1210  INPUT U$
1230  PRINT "ASIGNATURA"
1240  INPUT Y$
1250  FOR I=1 TO N
1260  IF U$=A$(I) THEN 1280
1270  GOTO 1350
1280  PRINT A$(I)
1290  FOR J=1 TO M
1300  IF Y $=Z$(J) THEN 1320
1310  GOTO 1340
1320  PRINT Z$(J), N$(I, J)
1330  GOTO 900
1340  NEXT J
1350  NEXT I
1360  END
```

Introducidos los apellidos y nombre de un alumno, se comparan con los almacenados en A$(I). Cuando se encuentra al alumno, se busca (instrucciones 1290-1340) la asignatura elegida.

Cuando se han cumplido ambas condiciones se imprime el nombre de la asignatura y las calificaciones del alumno en ella.

Ejercicio

16.2. Completar el programa anterior para que sea capaz de contar el número de *muy deficientes, insuficientes, suficientes, bienes, notables y sobresalientes* que tiene un alumno; y también, para que cuente el número de alumnos que han alcanzado estas calificaciones en cada asignatura.

Apéndice

La memoria externa más habitual (y más barata) en los pequeños microordenadores personales la constituye una casete de uso normal que permite almacenar programas y cierta cantidad no excesiva de datos.

En BASIC, para manejar este dispositivo, se utilizan las instrucciones Basic, SAVE y LOAD.

Si escribimos:

SAVE «ESTADISTICA»

y a continuación pulsamos la tecla [ENTER], se graba, en una cinta colocada en una casete en disposición de grabación, un programa al que llamamos ESTADISTICA, almacenado previamente en la memoria interna del microordenador.

De la misma manera, escribiendo

LOAD «ESTADISTICA» [ENTER]

un programa con el nombre «ESTADISTICA» almacenado en una cinta colocada en la casete (en disposición de reproducción) pasa a la memoria interna del microordenador.

Aunque estas instrucciones están ampliamente generalizadas admiten muchas variantes según el microordenador que se utilice.

Bibliografía

Programación Basic. JOHN G. KEMENY y THOMAS E. KURTZ. CECSA.

Basic Básico. Curso de Programación. RICARDO AGUADO y OTROS.

Programación de ordenadores en Basic. JESÚS SÁNCHEZ IZQUIERDO y FRANCISCO ESCRIHUELA VERCHER.

Basic. Programación de microordenadores. A. CHECROUN. Editorial Paraninfo.

Programación Basic. BYRON S. GOTTFRIED. *Colección Schaum.* McGraw-Hill.

Algunos programas de uso común en Basic. LON POOLE y MARY BORCHERS OSBORNE. McGraw-Hill.

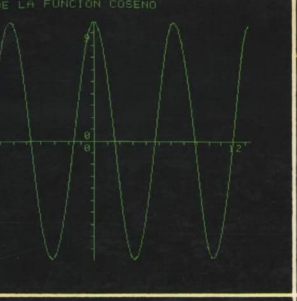
Lenguaje Basic. ROBERT L. ALBRECHT LE ROY FINKEL. Jerry Brown Limusa.

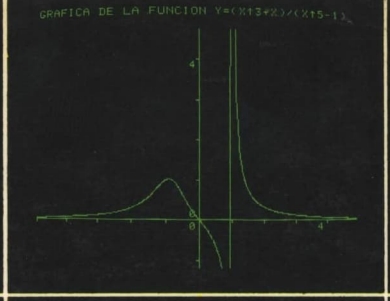
Computación I y II. The Open University. Curso Básico de matemáticas (unidades 8 y 20). McGraw-Hill.

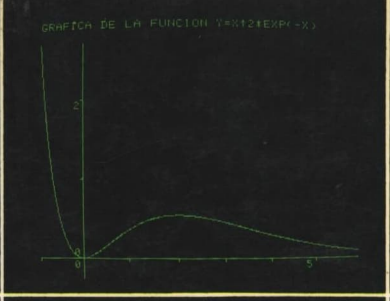
Le Basic Illustrè. DONALD ALCOCK. Editorial Masson.

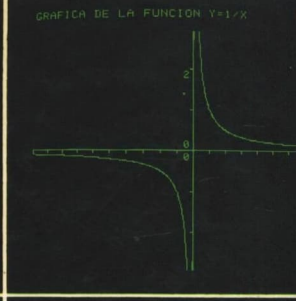
Analyse et programation en Basic. J. P. LAMOITIEN y OTROS. Bordas.

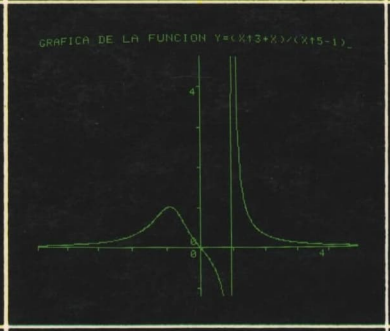
Comprendre la Micro Informatique. H. LILEN-J. P. NIERES. A Poly Classiques-Hachette.

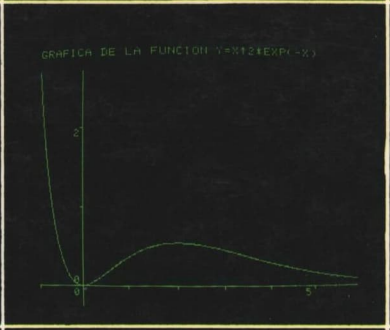


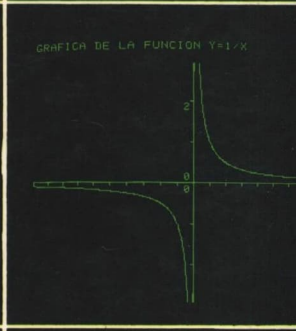


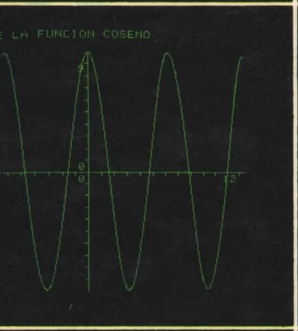


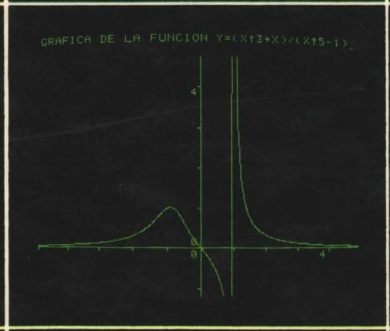


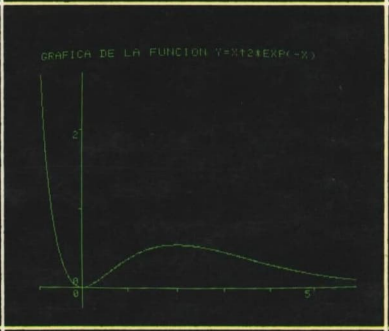


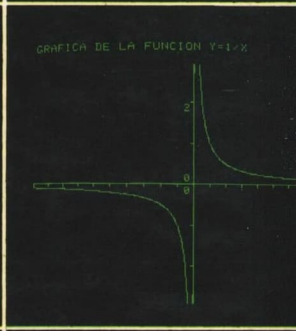


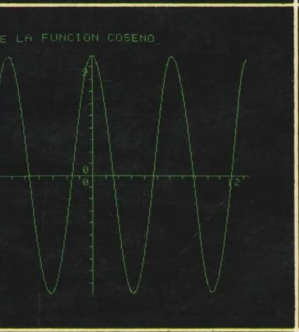


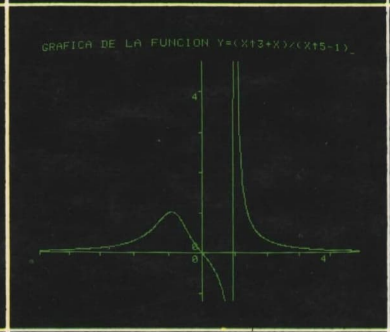


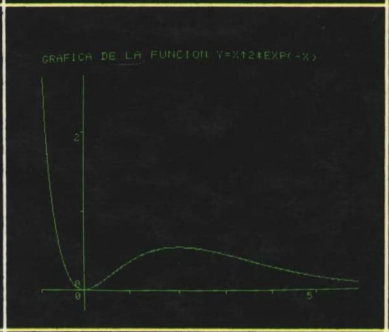


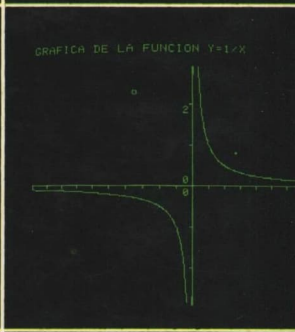


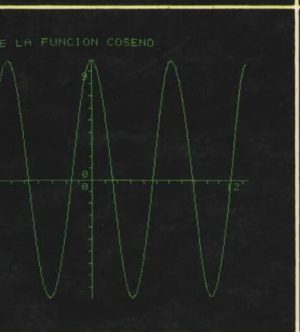


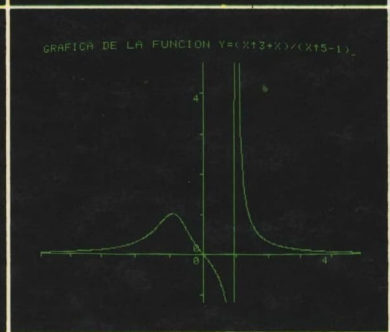


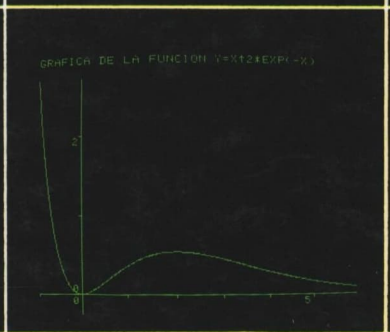


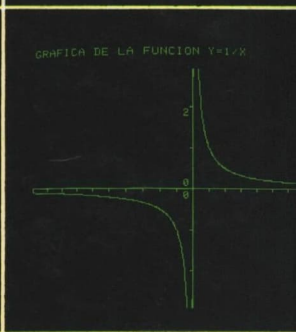


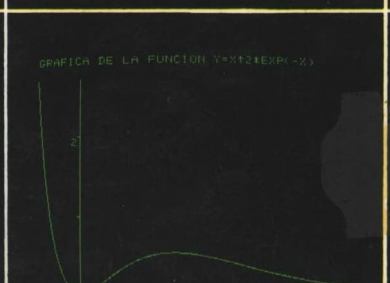


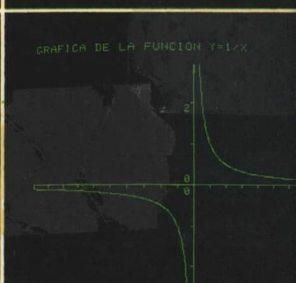












DISTRIBUIDOR EXCLUSIVO
COMERCIAL DE SM Ediciones
cesma, s.a.
Aguacate, 25 - MADRID-25